

用户手册

并行程序

版本：3.9

语言：简体中文



FLEXIV LTD.

www.flexiv.com

+86 400 8888 105 (中国大陆)

+886 2-7718-8868 (亚太地区)

+1 650-440-7808 (北美)

目录

1. 手册简介	4
2. 应用简介	5
3. 创建及管理并行程序	6
4. 编写并行程序	8
4.1 并行程序编辑界面	8
4.2 调试面板	9
5. 并行程序指令	11
5.1 指令类别	11
5.2 Signal 信号处理	12
5.2.1 定时等待	12
5.2.2 等待指定 I/O 端口状态	12
5.2.3 指定 I/O 端口赋值	12
5.2.4 指定 I/O 端口发送脉冲	13
5.2.5 获取 I/O 端口的值	13
5.3 Loops 循环控制	14
5.3.1 循环指定次数	14
5.3.2 满足条件时循环	14
5.3.3 遍历列表元素	15
5.3.4 跳出/继续 循环	16
5.4 Logic 逻辑运算	16
5.4.1 条件执行	16
5.4.2 比较	18
5.4.3 与/或	19
5.4.4 非	19
5.4.5 真/假	20
5.5 Math 数学运算	20
5.5.1 基本数字块	20
5.5.2 数学双目运算	20
5.5.3 浮点数转列表	21
5.5.4 列表转浮点数	21
5.6 Robot 机器人	22
5.6.1 抛出软件错误	22
5.6.2 清除软件错误	22
5.6.3 设置全局变量值	22

5.6.4 获取全局变量值	23
5.6.5 获取系统变量值	23
5.6.6 设置工具/工件坐标系的值	24
5.6.7 获取工具/工件坐标系的值	25
5.6.8 获取文件内容	25
5.6.9 写入文件内容	26
5.6.10 更新 AI 参数对象	26
5.6.11 清除 AI 参数对象	27
5.7 Text 文本处理	27
5.7.1 文本基本块	27
5.7.2 创建/拼接文本	28
5.7.3 获取文本长度	29
5.7.4 在文本字符串中查找子字符串	29
5.7.5 截取字符串	30
5.7.6 打印输出	31
5.7.7 数字和字符串转换	31
5.8 List 列表处理	32
5.8.1 创建列表	32
5.8.2 获取列表长度	33
5.8.3 获取列表元素值	34
5.8.4 设置列表元素值	35
5.8.5 列表和字符串转换	36
5.9 Communication 外部通讯	36
5.9.1 打开 Socket 连接	36
5.9.2 Socket 发送文本信息	37
5.9.3 Socket 接收文本信息	37
5.9.4 Socket 是否连接	38
5.9.5 关闭 Socket 连接	38
5.9.6 打开 Modbus TCP 连接	38
5.9.7 Modbus TCP 写入寄存器	39
5.9.8 Modbus TCP 读取寄存器	40
5.9.9 Modbus TCP 是否连接	40
5.9.10 关闭 Modbus TCP 连接	41
5.9.11 打开 Modbus RTU 连接	41
5.9.12 Modbus RTU 写入寄存器	41
5.9.13 Modbus RTU 读取寄存器	42
5.9.14 关闭 Modbus RTU 连接	43
5.9.15 打开 RS485 连接	43
5.9.16 RS485 发送数据	43
5.9.17 RS485 接收数据	44
5.9.18 关闭 RS485 连接	45

5.10 Variables 变量管理 ······	45
5.10.1 创建并行程序变量 ······	45
5.10.2 设置变量值 ······	46
5.10.3 变量自增 ······	46
5.10.4 获取变量值 ······	47
5.10.5 修改变量名 ······	47
5.10.6 删 除 变 量 ······	48
5.11 Functions 函数 ······	48
5.11.1 定义无返回值函数 ······	48
5.11.2 定义有返回值函数 ······	51
5.11.3 条件返回 ······	52
6. 运行并行程序 ······	53
6.1 启用/停用并行程序 ······	53
6.2 暂停/继续运行并行程序 ······	53
6.3 设置运行模式 ······	54
7. 查看并行程序状态 ······	55
7.1 查看并行程序运行状态 ······	55
7.2 查看并行程序运行详情 ······	56
7.3 查看并行程序日志 ······	57
7.4 查看导航栏状态 ······	57

1. 手册简介

本手册提供了并行程序应用程序的概述，并提供了具体和详细的说明，帮助您了解如何在 Flexiv Elements 中使用该应用。

本手册由以下章节组成：

第一章 手册简介

提供了本手册的简单介绍。

第二章 应用简介

介绍了并行程序。

第三章 创建及管理并行程序

介绍了如何在并行程序列表中创建及管理并行程序。

第四章 编写并行程序

提供了编程界面及调试面板概览。

第五章 并行程序指令

详细介绍了并行程序的指令类别及定义。

第六章 运行并行程序

介绍了如何启用/停用、暂停/继续并行程序、设置运行模式等。

第七章 查看并行程序状态

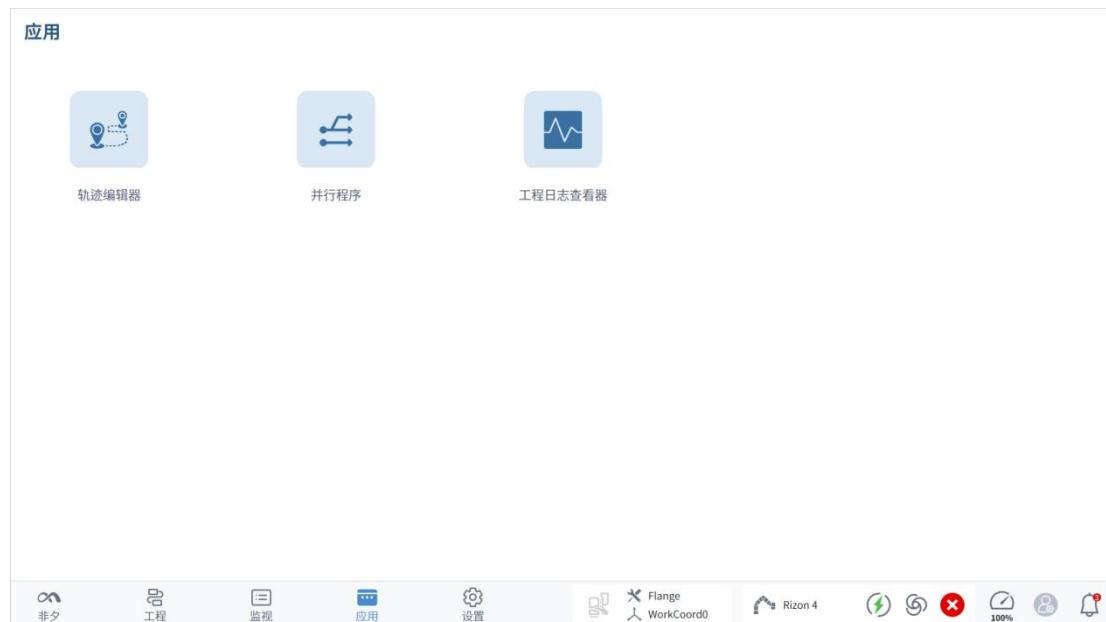
介绍了如何在底部导航栏中查看并行程序的状态。

2. 应用简介

并行程序是与机器人工程相互独立的逻辑处理程序。并行程序并不主动控制机器人的运动和力，但可以在机器人运行的同时实时监控信号、变量、位置、力等状态，并与外界设备保持实时的通讯。机器人最多可同时执行 5 个并行程序。并行程序可用于：

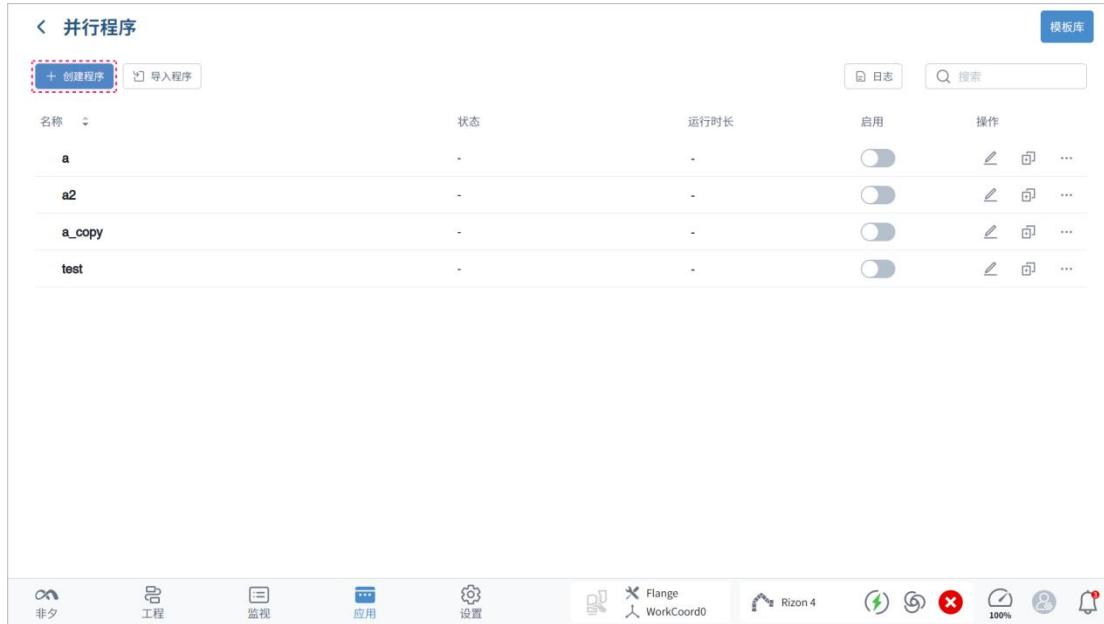
- **信号监控：**机器人运行时同步监控内部和外部信号的状态。
- **安全监控：**实时监控机器人安全状态并在出现异常时及时采取保护措施。例如：当机器人异常停机或者外力过大时，及时停止工具运行。
- **实时控制：**接收实时数据、实时修改变量值、实时输出信号。
- **通讯扩展：**可额外增加用户自定义的网络或串口通讯接口，与更多外部设备进行通讯。

您可以在此应用程序中创建、编写、运行和管理并行程序。要使用此应用，请前往 **Flexiv Elements > 应用 > 并行程序**。



3. 创建及管理并行程序

在并行程序列表界面，点击左上角 **创建程序** 即可创建一个新的并行程序。



The screenshot shows the 'Parallel Programs' management interface. At the top left is a back arrow and the title 'Parallel Programs'. On the right are buttons for 'Create Program' (highlighted with a red dashed box), 'Import Program', 'Logs', and 'Search'. Below is a table with columns: 'Name', 'Status', 'Run Time', 'Enabled', and 'Operations'. The table contains five entries: 'a', 'a2', 'a_copy', and 'test', all in an 'Enabled' state. At the bottom are various navigation and system status icons.

Name	Status	Run Time	Enabled	Operations		
a	-	-	<input checked="" type="checkbox"/>			
a2	-	-	<input checked="" type="checkbox"/>			
a_copy	-	-	<input checked="" type="checkbox"/>			
test	-	-	<input checked="" type="checkbox"/>			

您可以在并行程序列表界面中管理并行程序：

- 导入:** 点击 **导入程序** 按钮，从外部文件夹导入并行程序至列表中（列表中最多可有 20 个并行程序）
- 导入模板:** 点击 **模板库** 按钮，选择并导入要使用的内置并行程序模板至列表中
- 导出:** 点击 **导出** 按钮，将列表中的并行程序导出至外部文件夹中
- 编辑:** 点击 **编辑** 图标，可以进入并行程序编辑界面，编辑未启用的并行程序
- 查看:** 点击 **查看** 图标，可以查看已启用的并行程序的详细信息
- 创建副本:** 点击 **创建副本** 图标，将复制一份相同的并行程序至列表中
- 重命名:** 点击 **重命名** 按钮，可以修改未启用的并行程序的名称
- 删除:** 点击 **删除** 按钮，删除未启用的并行程序

< 并行程序

+ 创建程序 导入程序

名称	状态	运行时长	启用	操作
a	-	-	<input checked="" type="checkbox"/>	
a2	-	-	<input checked="" type="checkbox"/>	
a_copy	-	-	<input checked="" type="checkbox"/>	
test	-	-	<input checked="" type="checkbox"/>	

日志 搜索

模板库

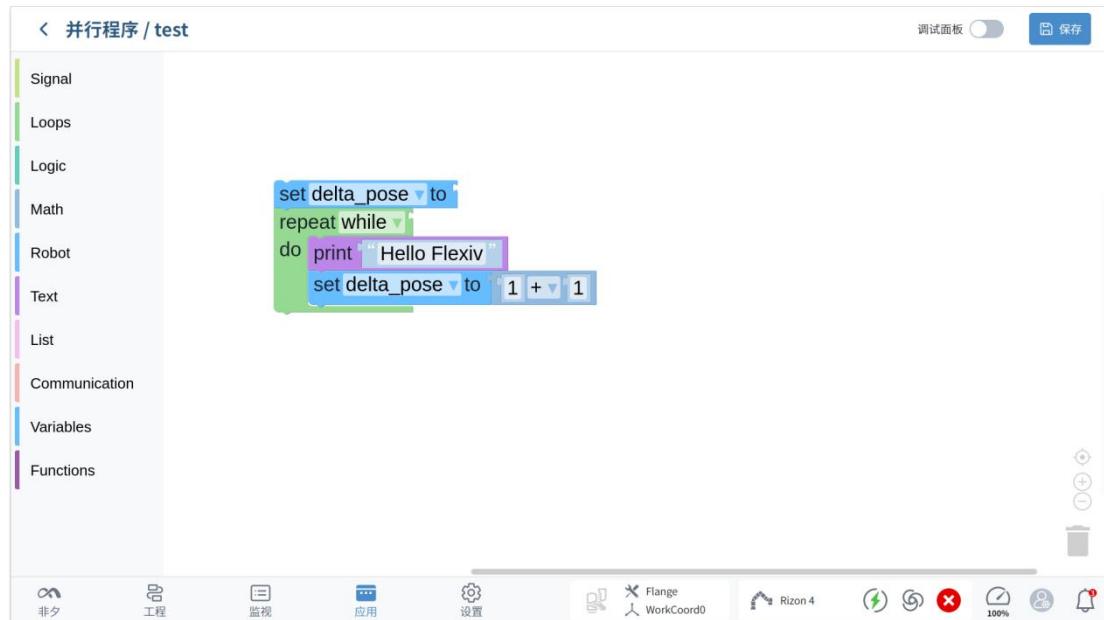
导出 重命名 删除

非夕 工程 监视 应用 设置 Flange WorkCoord0 Rizon 4 100%

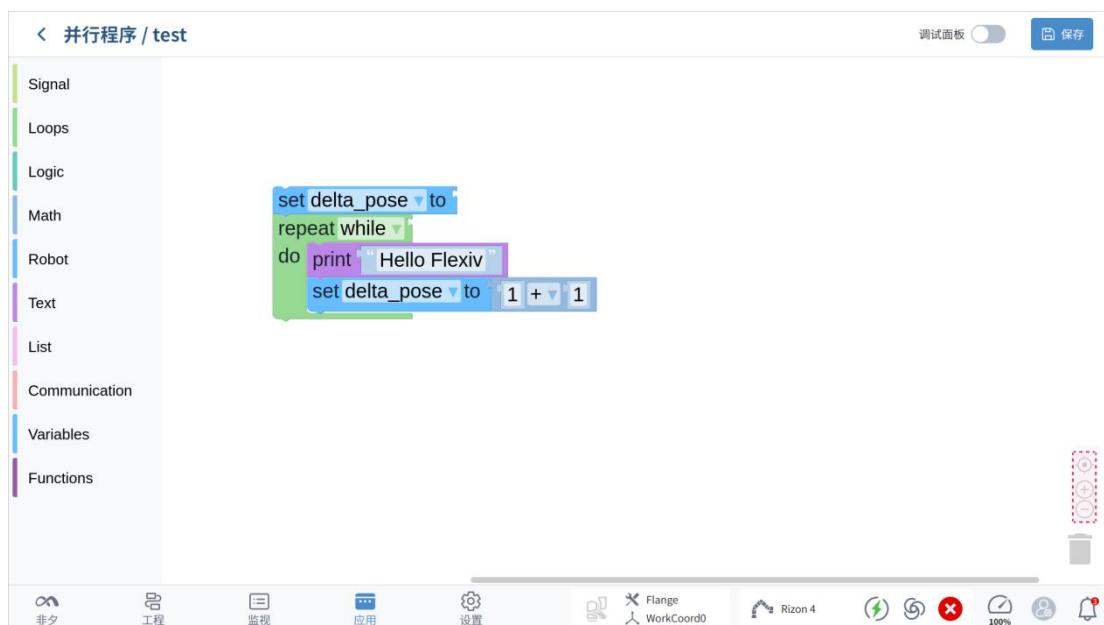
4. 编写并行程序

4.1 并行程序编辑界面

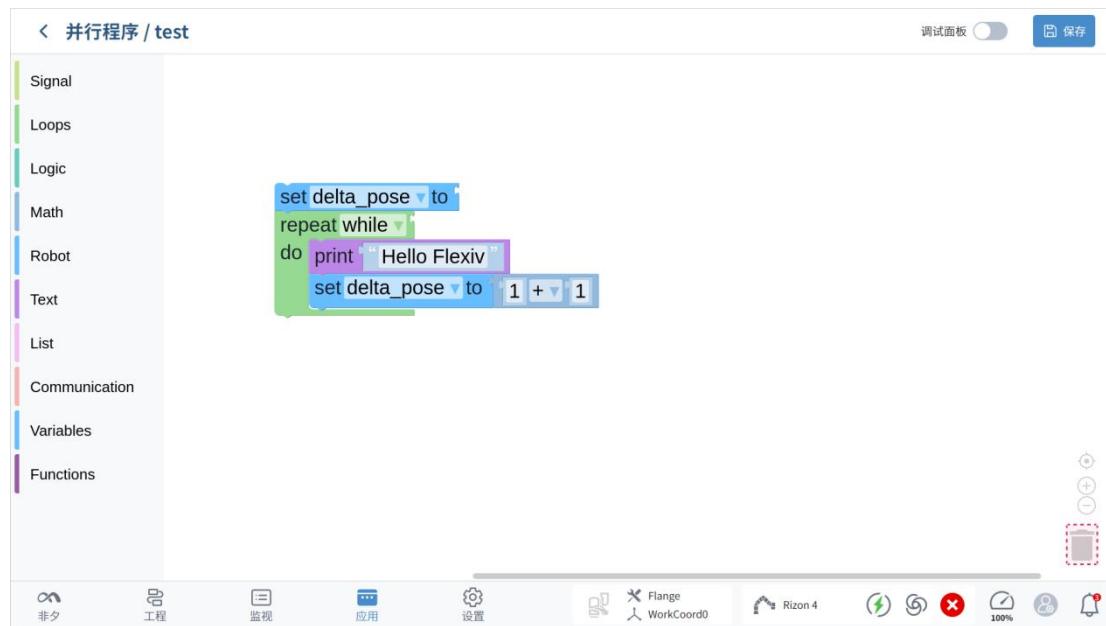
并行程序由图形化的指令拼接而成。在并行程序编辑界面中，您可以从左侧的指令列表中拖拽出并行程序指令，快速编辑和拼接指令完成并行程序的编写。



您可以点击缩放控件以缩放显示工作区。

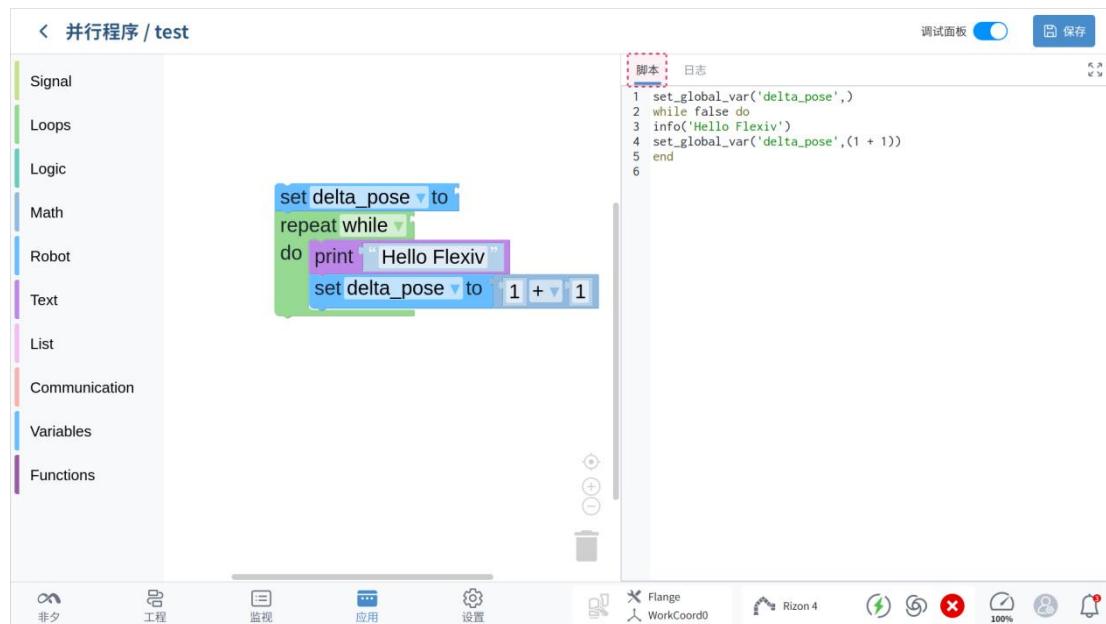


若要删除指令块，可将其拖放至右下角的垃圾桶中。您还可以点击垃圾桶，查看已删除的指令块并将其恢复。



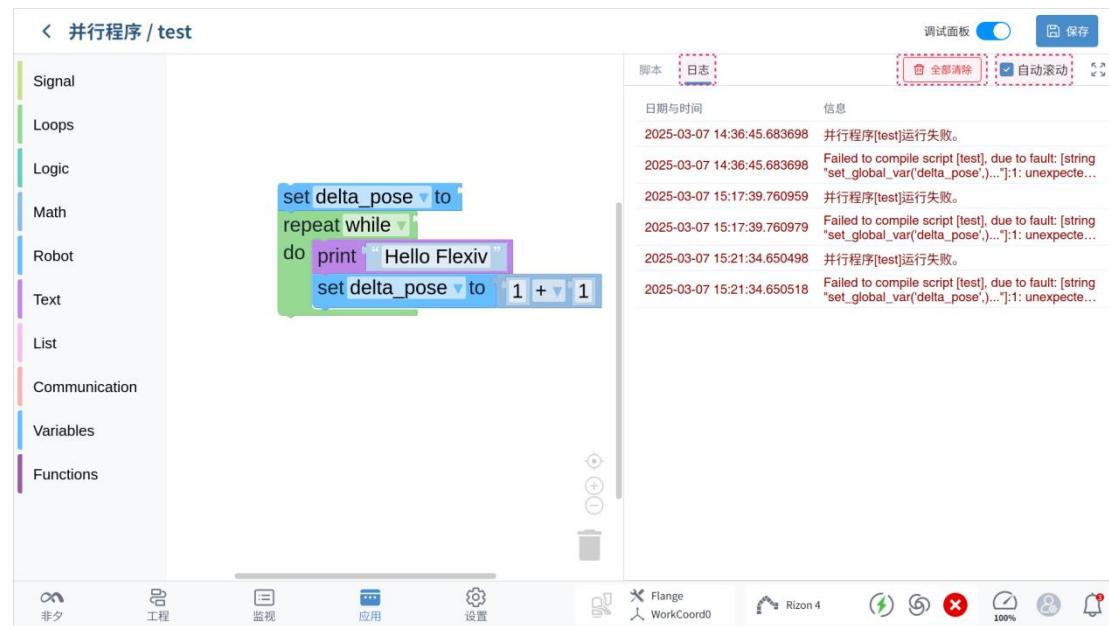
4.2 调试面板

点击 **调试面板** 显示开关，可查看当前并行程序的脚本及日志。点击 **脚本** 选项卡，可以查看图形化的指令的文字脚本形式。



点击 **日志** 选项卡，可以查看当前并行程序的日志。您可以：

- **全部清除：** 清除当前并行程序的所有日志
- **自动滚动：** 并行程序日志将自动向下滚动

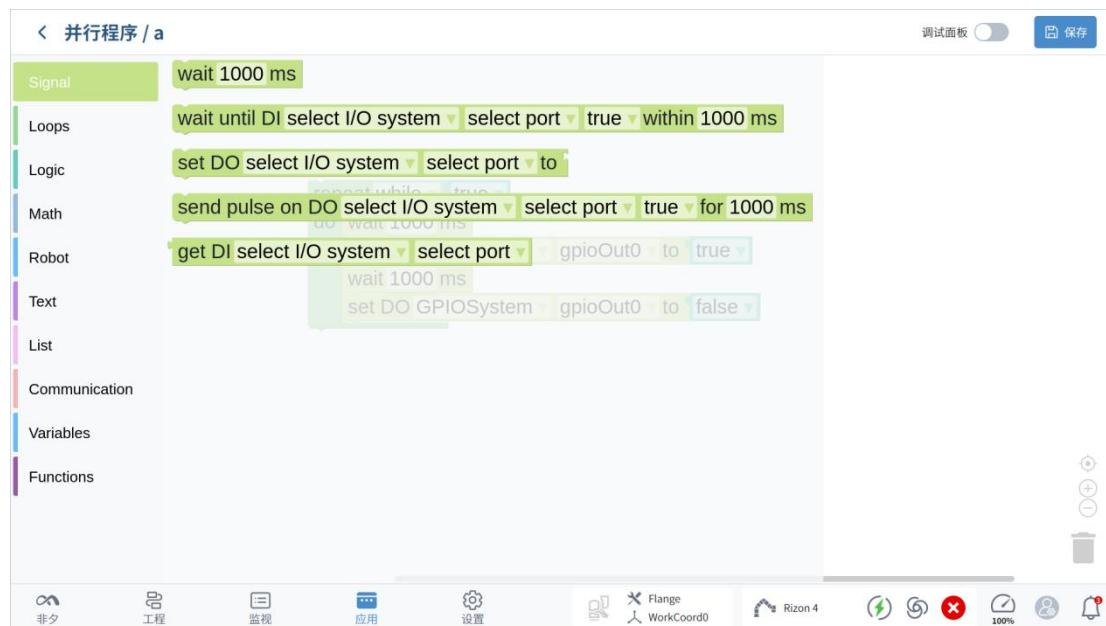


编写完成后，点击右上角的 **保存** 即可保存并行程序。

5. 并行程序指令

5.1 指令类别

并行程序提供了丰富的图形化指令，点击指令类别，可展开显示该类别下的所有指令。



并行程序指令的类别如下：

类别	定义	指令
Signal	信号处理	等待时间/信号；设置输出信号；发送脉冲；获取信号
Loops	循环控制	循环固定次数；循环直到满足条件；列表元素循环；中断循环
Logic	逻辑运算	条件运行；数值比较；与/或/非；真/假
Math	数学运算	新建数值；数值运算；浮点与字节间转换
Robot	机器人	抛出/清除错误；获取/设置全局变量值；获取系统变量值；读写工具/工件坐标系；读写系统文件；更新目标物体状态
Text	文本处理	新建文本；获取文本长度；找到子文本位置；截取子文本；打印文本；字符与数字间转换
List	列表处理	新建列表；获取列表长度；获取/设置列表元素值；列表与文本转换
Communication	外部通讯	打开/关闭 socket / Modbus TCP / Modbus RTU / RS485 通讯；发送数据；接收数据
Variables	变量管理	新建并行程序变量；获取/设置变量值；更改变量值
Functions	函数	新建函数；调用函数

5.2 Signal 信号处理

5.2.1 定时等待

`wait 1000 ms`

- 功能描述

使并行程序任务阻塞，并等待一定的时间。该时间由用户指定，单位为毫秒。

5.2.2 等待指定 I/O 端口状态

`wait until DI select I/O system ▼ select port ▼ true ▼ within 1000 ms`

- 功能描述

- 用于监测指定的 I/O 系统端口在设定的时间内，其指定索引的端口值是否变为预期的值。
这通常用于等待外部设备或传感器的状态改变，如检测 GPIO 端口信号或其他类型的接口信号。

- 参数说明：

参数 1：指定要监控的 I/O 系统端口的名称。

参数 2：端口的具体索引，用于指定在复合端口设备中具体的子端口号。

参数 3：期望端口值变为该值。具体类型取决于端口支持的数据类型。

参数 4：等待时间，单位为毫秒。在此时间段内，函数将监测端口值是否达到指定的值。
当设置为 0 时，保持等待。

- 用法示例

a. **选择 I/O 系统：**选择 I/O 系统，如 `GPIOSystem`。

b. **选择 I/O 端口：**选择 I/O 端口，如 `gpioln1`。

c. **选择条件：**选择预期的值，比如为 `true`。

d. **选择超时：**设置最大的等待时间，如 `1000ms`。当超过该时间，就算没有到达预期值，该块也会停止阻塞。

`wait until DI GPIOSystem ▼ gpioln1 ▼ true ▼ within 1000 ms`

- 运行结果

将保持阻塞，直到 `GPIOSystem` 端口的 `gpioln1` 为 `true`，最多阻塞 1000 毫秒。

5.2.3 指定 I/O 端口赋值

`set DO select I/O system ▼ select port ▼ to []`

- 功能描述

此功能块用于设置指定的 I/O 系统端口的值。

- 用法示例
 - a. 选择 I/O 系统：选择 I/O 系统，如 GPIOSystem。
 - b. 选择 I/O 端口：选择 I/O 端口，如 gpioOut1。
 - c. 添加预期值：添加一个值作为该端口的预期设定值，如 true。

```
set DO GPIOSystem ▾ gpioOut1 ▾ to true ▾
```

- 运行结果

GPIOSystem 端口的 gpioOut1 会被设置为 true。

5.2.4 指定 I/O 端口发送脉冲

```
send pulse on DO select I/O system ▾ select port ▾ true ▾ for 1000 ms
```

- 功能描述

此功能块用于朝指定 I/O 系统端口发送一个脉冲信号。

- 用法示例

- a. 选择 I/O 系统：选择 I/O 系统，如 GPIOSystem。
- b. 选择 I/O 端口：选择发送脉冲波的 I/O 端口，如 gpioOut1。
- c. 选择脉冲类型：当此参数为 true 时，发送正脉冲；为 false 时，发送负脉冲。选择 true 发送一个正脉冲波。
- d. 持续时间：选择脉冲的持续时间，单位是毫秒。

```
send pulse on DO GPIOSystem ▾ gpioOut1 ▾ true ▾ for 1000 ms
```

- 运行结果

将会向 GPIOSystem 的 gpioOut1 端口发送一个持续 1000 ms 的脉冲波。

5.2.5 获取 I/O 端口的值

```
get DI select I/O system ▾ select port ▾
```

- 功能描述

此功能块用于获取指定的 I/O 系统端口的值。

- 用法示例

- a. 选择 I/O 系统：选择 I/O 系统，如 GPIOSystem。
- b. 选择 I/O 端口：选择要获取的 I/O 端口，如 gpioIn1。

```
get DI GPIOSystem ▾ gpioIn1 ▾
```

- 运行结果及返回值

返回一个 bool 类型，表示 GPIOSystem 的 gpioIn1 端口的值。

5.3 Loops 循环控制

5.3.1 循环指定次数



- 功能描述

循环一定次数的功能框中的指定操作。

- 用法示例

- a. 输入指定次数
- b. 添加指定操作



- 运行结果

并行程序日志中打印十次 “Hello flexiv!”。

5.3.2 满足条件时循环

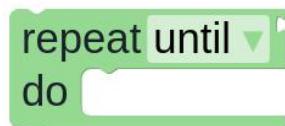
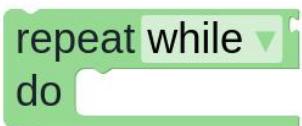


- 功能描述

当满足一定条件时，循环做功能框内的指定操作。

- 用法示例

- a. 选择循环逻辑类型：比如 while 或者 until。



- b. 添加循环条件：比如 true 或 false。

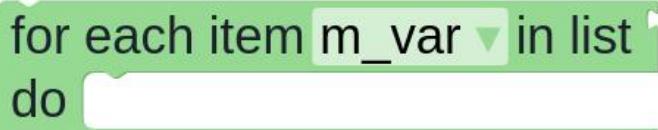
- c. 添加指定操作



- 运行结果

并行程序日志中持续打印“Hello flexiv!”。

5.3.3 遍历列表元素



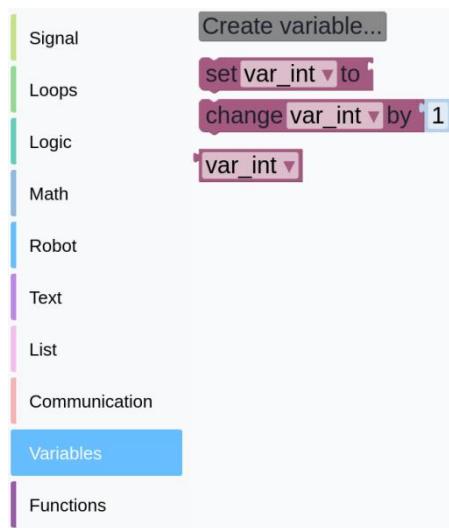
- 功能描述

此功能用于遍历列表中的每个元素，对列表中的每个项目执行指定的操作。它是编程中常见的迭代结构，允许用户对数据集合中的每一项进行处理，例如进行计算、修改或条件判断。

- 用法示例

a. **指定列表:** 在功能块中，首先需要指定想要遍历的列表。可以是一个变量，或者直接输入 的列表数据。

b. **临时变量:** 指定每次遍历的临时变量名，默认为 var_int，该临时变量可以在 **变量** 一栏找到。



a. **执行操作:** 为每个列表中的元素定义要执行的操作。这个操作可以是任何有效的代码块，如计算表达式、函数调用或其他控制流语句。

b. **遍历执行:** 功能块将自动逐一处理列表中的每个元素，按照设定的操作执行。



- 运行结果

运行后会依次输出“1”“2”“3”。

5.3.4 跳出/继续 循环

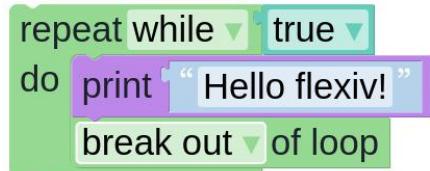
break out ▾ of loop

- 功能描述

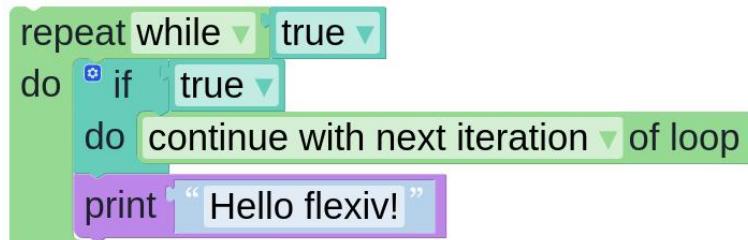
跳出或继续循环。

- 用法示例

- 该功能块无法单独使用，必须配合其它循环块使用，比如跳出 repeat 循环。



- 或者继续 repeat 循环，并跳过该功能后面的功能块（在循环框内的）。同样，该功能块一般配合 if do 功能块使用，当达到某一条件时，跳过后面的功能块，直接进入下一轮循环。



5.4 Logic 逻辑运算

5.4.1 条件执行

**if
do**

- 功能描述

当条件成立时，运行功能框内的语句。

- 用法示例 1

- 添加成立条件。
- 添加要执行的逻辑块。

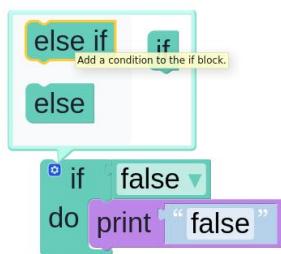


- 运行结果及返回值

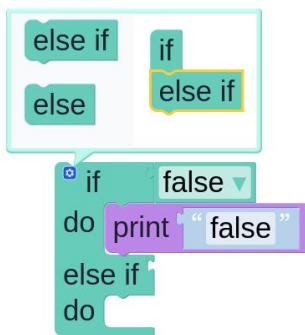
运行后会输出：“Hello flexiv!”。

- 用法示例 2

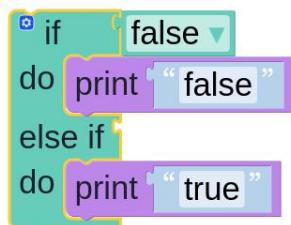
- 添加成立条件。
- 添加要执行的逻辑块。
- 点击左上角的 **设置** 图标。



- 将需要的配置拖动到 if 的下面，比如 **else if**。

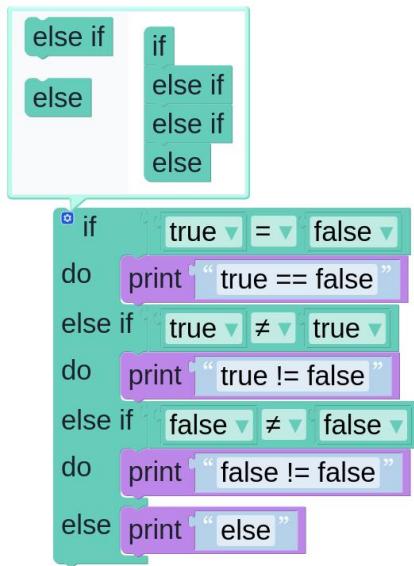


- 再次点击 **设置** 图标，确认该配置。继续添加想新的成立条件。



- 用法示例 3

a. 和上述步骤类似，可以添加多个 **else if** 以及最多一个 **else**。



5.4.2 比较



- 功能描述

双目逻辑运算符，用于比较两个相同类型的值。

- 用法示例

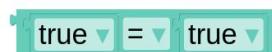
a. 将两个返回相同类型值的块添加到该功能块的两个空缺中。

b. 选择运算符。

◦ **是否等于：**两个值相等时，返回真，否则返回假。



结果返回真。



结果返回真。

◦ **是否不等于：**两个值不相等时，返回真，否则返回假。



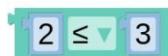
结果返回假

◦ **是否小于：**前一个值小于后一个值时，返回真，否则返回假。



结果返回真

- **是否小于等于:** 前一个值小于等于后一个值时，返回真，否则返回假。



结果返回真

- **是否大于:** 前一个值大于后一个值时，返回真，否则返回假。



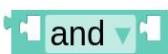
结果返回假

- **是否大于等于:** 前一个值大于等于后一个值时，返回真，否则返回假。



结果返回假

5.4.3 与/或



- **功能描述**

双目运算符，用来计算逻辑与、逻辑或。

- **用法示例**

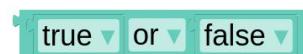
- 将两个逻辑块添加到该功能块的两个空缺中。

- 选择运算符

逻辑与: 两个值都为真时，返回结果为真。



逻辑或: 两个值至少有一个为真时，返回结果为真。



- **运行结果及返回值**

该功能块会返回一个 `bool` 类型。以上两个示例中将分别返回：`false` 和 `true`。

5.4.4 非



- **功能描述**

单目运算符，可以用来计算逻辑非。

- **用法示例**

- 将逻辑语句块或者基本逻辑块，添加到该功能块的后面。比如，将逻辑真添加到该功能块。



- 运行结果及返回值

整个逻辑块组合将返回一个 `bool` 类型 `false`。

5.4.5 真/假



- 功能描述

逻辑真或逻辑假。

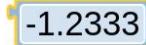
5.5 Math 数学运算

5.5.1 基本数字块



- 功能描述

基本数字功能块，可以填写一个 `Number` 类型，即整数、小数、负数。

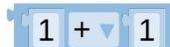


5.5.2 数学双目运算

- 功能描述

该功能块需要两个数字作为参数，包含一系列数字运算的集合，包括：

加法： $A + B$



减法： $A - B$



乘法： $A \times B$



除法： $A \div B$



次方： $A ^ B$



5.5.3 浮点数转列表

make bytes from float 0

- 功能描述

- Float 是单精度浮点数，它需要四个字节来表示。您可以使用该功能块将一个浮点数转换为四个字节的列表。
- 假设浮点数按照阅读序，从高到低位分别为：ABCD，A 代表最高位，D 代表最低位。转换为列表后，存储为顺序为：{[1] = A, [2] = B, [3] = C, [4] = D}。

- 用法示例

- 添加一个任意数字

make bytes from float 1.31

- 运行结果及返回值

返回结果是一个数组，包含四个数字：{[1] = 63, [2] = 167, [3] = 174, [4] = 20}。

5.5.4 列表转浮点数

make float from bytes 0 0 0 0

- 功能描述

- 将一个包含 4 个数字的数组列表转换为一个浮点数。
- 假设预期的浮点数按照阅读序，从高到低位分别为：ABCD，A 代表最高位，D 代表最低位。那么应该填写到该功能块的顺序也应该为：{[1] = A, [2] = B, [3] = C, [4] = D}。

- 用法示例

- 通常，用户不方便自己去构造四个数字。所以，可以使用前面【浮点数转列表】功能块的结果来构造。
- 先使用【浮点数转列表】生成一个列表。这里用到了变量的内容，可以提前熟悉一下变量的使用。

set bytes_var to make bytes from float 1.31

- 然后需要用到【列表取值】功能块来获取该数组的第一个到第四个元素。

```
set bytes_var to make bytes from float 1.31
print make float from bytes in list [bytes_var get # 1] in list [bytes_var get # 2] in list [bytes_var get # 3] in list [bytes_var get # 4]
```

- 运行结果及返回值

该功能块返回一个浮点数，并且这个浮点数的值应该等于示例中输入的“1.31”。

5.6 Robot 机器人

5.6.1 抛出软件错误

`prompt fault` 

- 功能描述

触发一个软件层错误。

- 用法示例

- 在文本框中添加错误信息字符串。

`prompt fault "This is a test fault info"`

- 运行结果

弹窗报错，并且当前并行程序任务触发软件错误状态。

5.6.2 清除软件错误

`clear fault`

- 功能描述

清除已有的软件错误。

- 运行结果

系统的错误状态被清除。

5.6.3 设置全局变量值

`set select global variable to`

- 功能描述

给机器人系统的一个全局变量赋值。

- 用法示例

- a. **创建全局变量：** 使用该块的前提是已经在机器人系统中添加了一个全局变量。



- b. **选择全局变量：** 选择要设置的全局变量的名称。

set select global variable ▾ to

select global variable
test_global_var

- c. **添加值块:** 添加要为全局变量赋值的一个值。该值的类型以及长度，要和全局变量的赋值要求的类型及长度保持一致。否则，会导致失败。

set test_global_var ▾ to 10

◦ **运行结果**

全局变量 `test_global_var` 的值被赋值为 10。

5.6.4 获取全局变量值

select global variable ▾

- **功能描述**

读取机器人系统的一个全局变量的值。

- **用法示例**

- a. **创建全局变量:** 使用该块的前提是已经在机器人系统中添加了一个全局变量。



- b. **选择全局变量:** 选择要读取的全局变量的名称。

select global variable ▾

select global variable
test_global_var

- c. **使用指令块:** 将指令块与其他指令块拼接。

print test_global_var ▾

◦ **运行结果**

打印出全局变量 `test_global_var` 的值。

5.6.5 获取系统变量值

select system variable ▾

- **功能描述**

读取机器人系统的一个系统变量的值。

- 用法示例

- 选择系统变量：**选择要读取的系统变量的名称。



- 使用指令块：**将指令块与其他指令块拼接。

print CartesianForceZ

- 运行结果

打印出机器人 TCP 在 Z 方向上受到的力。

5.6.6 设置工具/工件坐标系的值

set Work coordinate system ▾ select work coordinate system ▾ to ▾

- 功能描述

设置工具或者工件坐标系的值。

- 用法示例

- 选择 Work coordinate system 或者 Tool。**
- 选择一个工件或者工具名称。**
- 创建一个长度为 6 的列表，并将它添加到此块后面。**



- 运行结果及返回值：运行这两个块后，相应的工件 - WorkCoord0 以及工具 - Flexiv-GN01 的坐标值应该被设定为了相应数据（注意：列表的值和坐标系值之间有换算关系）。

The screenshot shows a CAD software interface with two coordinate system definitions:

WorkCoord0			
X	1000.00	Rx	4.00
Y	2000.00	Ry	5.00
Z	3000.00	Rz	6.00

Flexiv-GN01					
TCP	位置 [mm]	X	1000.00	Rx	4.00
		Y	2000.00	Ry	5.00
	欧拉角 [deg]	Z	3000.00	Rz	6.00

5.6.7 获取工具/工件坐标系的值

Work coordinate system ▼ select work coordinate system ▼

- 功能描述

获取工件或者工具坐标系的值。

- 用法示例

- 选择 Work coordinate system 或者 Tool。
- 选择一个工件或工具名称。
- 打印输出。

print Work coordinate system ▼ WorkCoord0 ▼

print Tool ▼ Flexiv-GN01 ▼

- 运行结果及返回值：运行以上两个块，都会返回一个长度为 6 的列表，其结果分别为：

```
{[1] = 0.001,[2] = 0.00223,[3] = 0.00012,[4] = 4.11,[5] = 0,[6] = 0,}
```

```
{[1] = 0.001,[2] = 0.002,[3] = 0.5,[4] = 0,[5] = 0,[6] = 0,}
```

5.6.8 获取文件内容

content of file trajectory ▼ “input the file name”

- 功能描述

指定一个要读取的文件类型以及文件名，该块会从该文件类型的默认文件夹下，读取指定文件名的文件内容。

- 用法示例

- a. 选择要读取的文件类型，比如当前默认的 trajectory。
- b. 输入文件名。注意，文件名不需要后缀。

```
content of file trajectory ▾ "my_test_traj_file"
```

- 运行结果及返回值

以字符串的形式返回整个文件内容。

5.6.9 写入文件内容

```
write content " " to trajectory ▾ file "input the file name"
```

- 功能描述

指定一个要写入的文件类型以及文件名，该块会从该文件类型的默认文件夹下，将指定内容写入该文件。

- 用法示例

- a. 添加要写入的字符串。
- b. 选择要读取的文件类型，比如当前默认的 trajectory。
- c. 输入文件名。注意，文件名不需要后缀。

```
write content "traj_coord_type: ""\ntraj_coord_ref: ""\npattern... " to trajectory ▾ file "my_test_traj_file_2\"
```

- 运行结果及返回值

成功：写入指定内容到指定文件。

失败：未写入。

5.6.10 更新 AI 参数对象

```
update object name "input the object name" type ARRAY_VEC_2D ▾ data
camera intrinsics "1378.57 1378.57 1250 1250 2500 2500 1"
extrinsics "0 0 0 0 0" mounting flange ▾
```

- 功能描述

- 更新 AI 参数对象的值。该参数对象由一组相关的参数组成的，具体如下。
- 参数说明

参数 1：对象的名称，唯一标识一个对象。

参数 2：对象的类型标识：指定对象的数据类型或用途。

参数 3：对象的值，具体内容取决于对象的类型。

参数 4：描述相机的内部光学特性，如焦距和主点。

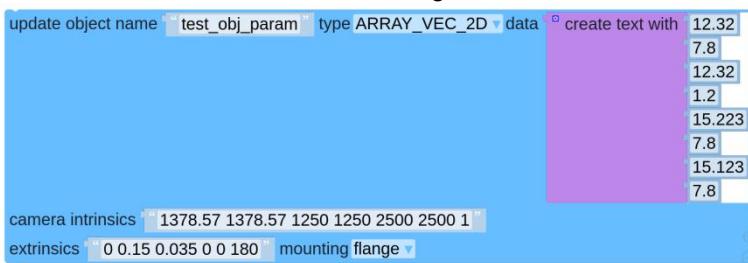
参数 5：描述相机相对于世界坐标系统的位置和朝向。

参数 6：描述对象在场景图中的层级关系或连接关系。

- 用法示例

此次示例的场景是：用相机来对目标物体进行标定，并通过标定结果来创建一个视觉伺服参数对象。

- 输入名称：**输入 AI 参数对象的名称。如果该名称的对象不存在，则创建一个。
- 选择参数类型：**选择一个参数类型，表示要希望以什么类型来更新 AI 参数对象。这里选择 **ARRAY_VEC_2D**。
- 添加属性值：**添加一个列表作为该 AI 参数的属性值，该值在不同的场景中有不同的含义。一般来说，这个值代表一组坐标，用来确定一个物体在坐标系中各个顶点的坐标。在此次示例中，此参数是四组二维坐标，用来确定标定物（矩形）的四个顶点。
- 构造相机内参：**以空格作为分隔填入第四个参数文本框中。
- 构造相机外参：**相机外参是和最后一个参数，即对象的父链接相关的。在不同的父链接中有不同的含义，这里的外参分别是相机相当于机械臂的相对位置以及欧拉角。
- 选择对象的父链接：**选择父链接 **flange** 操作。



- 运行结果及返回值

不返回任何值。如果发生错误，可能会触发软件错误，使并行程序停止运行。如果要查看更新后的参数状态，可以前往 [工程](#) 界面查看。

5.6.11 清除 AI 参数对象

clear object pool

- 功能描述

清除当前已有的 AI 参数对象。

5.7 Text 文本处理

5.7.1 文本基本块



- 功能描述

基本的文本块，可以直接在功能块中输入任意文本信息。但是，当前不支持多行文本输入，仅支持单行。

5.7.2 创建/拼接文本

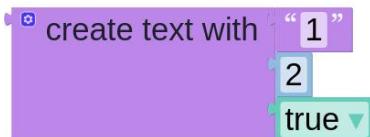
• create text with

- 功能描述

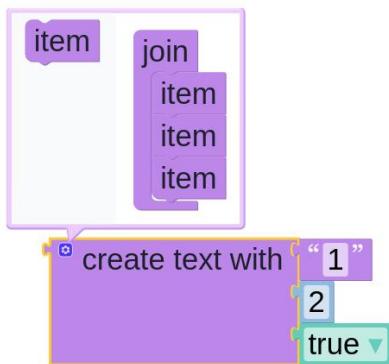
基于已有文本，创建一个新的文本。已有文本可以是一个或者多个。也可以是 0 个，即代表创建一个空文本。

- 用法示例

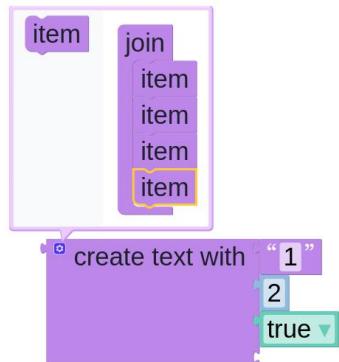
- 给后面的空缺添加值块，可以是任意类型变量。



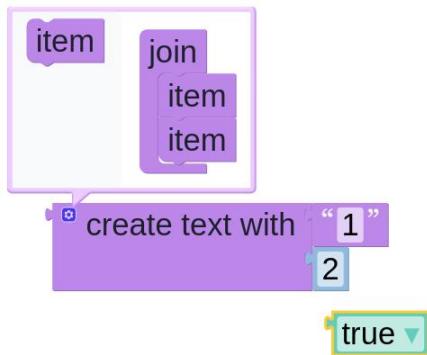
- 点击齿轮图标，可以设置缺省块的数量。



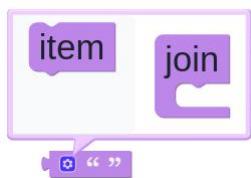
- 想要几个元素就拖入几个 item 到右边的 join 下。



- 要减少元素也是一样，从 join 下拖走对应的 item 即可。



- 如果要创建一个空字符串，将所有 item 都移除即可。



- 运行结果及返回值

返回一个文本。

5.7.3 获取文本长度

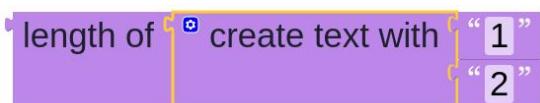
length of “abc”

- 功能描述

用于获取文本内容的长度。

- 用法示例

- 添加字符串：将要访问的字符串拖入到此块中（示例中是临时创建了一个字符串）。



- 运行结果及返回值

返回结果为一个 number 类型 2，表示字符串的长度。

5.7.4 在文本字符串中查找子字符串

in text **text** **find first occurrence of text** “abc”

- 功能描述

该块是对字符串查找操作的集合，其中包含了：

找到子字符串第一次在主字符串中出现的位置

找到子字符串最后一次在主字符串中出现的位置

- **用法示例 1:** 找到子字符串第一次在主字符串中出现的位置
 - a. **添加字符串:** 将要访问的字符串拖入到此块中（示例中是临时创建了一个字符串）。
 - b. **选择位置:** 选择 `first`。
 - c. **添加子字符串:** 添加目标子字符串。
 - **运行结果及返回值**

返回一个 `number` 类型，表示子字符串 “abc” 第一次在主串 “abcabc” 出现的位置，即 1。
- **用法示例 2:** 找到子字符串最后一次在主字符串中出现的位置
 - a. 同上
 - b. **选择位置:** 选择 `last`。
 - c. 同上
 - **运行结果及返回值**

返回一个 `number` 类型，表示子字符串 “abc” 最后一次在主串 “abcabc” 出现的位置，即 4。

5.7.5 截取字符串

`in text [text] get substring from letter # [1] to letter # [3]`

- **功能描述**
根据指定起始下标和结束下标，获取字符串的子字符串。
- **用法示例 1:** 手动指定起始和结束下标
 - a. **添加字符串:** 将要访问的字符串拖入到此块中（示例中是临时创建了一个字符串）。
 - b. **选择操作:** 起始和结束模式，都选择 `letter#` 操作。
 - c. **添加起始和结束下标值块:** 手动添加两个下标到 `letter#` 号后面。

`in text [create text with "ab" get substring from letter # [1] to letter # [3]] "cd"`

- **运行结果及返回值**
返回字符串 “abcd” 的第 1 个到第 3 个之间的子字符串 “abc”。
- **用法示例 2:** 手动指定方向的起始下标和结束下标
 - a. **添加字符串:** 同上
 - b. **选择操作:** 起始和结束模式，都选择 `letter# from end` 操作。
 - c. **添加起始和结束下标值块:** 同上

```
print in text ⚡ create text with “ab” get substring from letter # from end ▾ 3 to letter # from end ▾ 1
      “cd”
```

- 运行结果及返回值

返回字符串“abcd”的倒数第3个到倒数第1个之间的子字符串“bcd”。

- **用法示例 3:** 同样，可以直接指定 **first letter** 或者 **last letter** 的方式来获取

```
print in text ⚡ create text with “ab” get substring from first letter ▾ to last letter ▾
      “cd”
```

- 运行结果及返回值

返回字符串“abcd”的第1个到最后一个之间的子字符串“abcd”。

5.7.6 打印输出

```
print “”
```

- **功能描述**

打印输出到屏幕。

- 可以添加字符串到 **print** 的参数块。

```
print “1234”
```

- 也可以添加任意类型的值到参数块，如数字。

```
print 1
```

或者逻辑。

```
print true ▾
```

5.7.7 数字和字符串转换

```
make number from text ▾ in format DEC ▾
```

- **功能描述**

- 将数字转换为指定的进制字符串。
- 以指定的进制将字符串解析为数字。

- **用法示例 1:** 字符串转数字

- a. **选择操作:** 选择 **number from text** 操作。
- b. **添加待转换值块:** 添加一个字符串块到此块上，确保该字符串中的值，对应后续的进制，是一个有效值。
- c. **选择进制:** 选择希望以什么进制来解析该字符串值。

```

make number from text "10" in format DEC ▾
make number from text "10" in format HEX ▾
make number from text "10" in format OCT ▾
make number from text "10" in format BIN ▾

```

- 运行结果及返回值

如上所示四个块，返回的都是 **number** 类型。如果以十进制来看待这些返回的数字，它们的值分别是：10、16、8、2。

- **用法示例 2：** 数字转字符串

- a. **选择操作：** 选择 **text from number** 操作。
- b. **添加待转换值块：** 添加一个数字块到此块上。这个数字块中的值是以 10 进制来看待的。
- c. **选择进制：** 选择希望以什么进制来存储这个数字。

```

make text from number 10 in format DEC ▾
make text from number 16 in format HEX ▾
make text from number 8 in format OCT ▾
make text from number 2 in format BIN ▾

```

- 运行结果及返回值

如上所示四个块，返回的都是字符串类型，并且都是“10”。

5.8 List 列表处理

5.8.1 创建列表

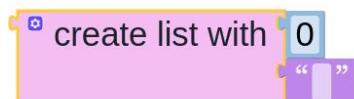
 **create list with**

- 功能描述

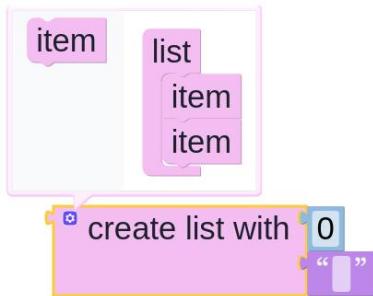
创建一个新的列表，并赋值。如果赋值块数量为空（可以手动设置赋值块数量），即代表创建一个空列表。

- **用法示例**

- 给后面的空缺添加值块，可以是任意类型的变量。

 **create list with** 0

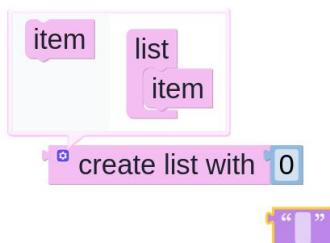
- 点击齿轮图标，可以设置缺省块的数量。



- 想要几个元素就拖入几个 item 到右边的 list 下。



- 要减少元素也是一样，从 list 下拖走对应的 item 即可。



- 如果要创建一个空列表，将所有 item 都移除即可。



- 运行结果及返回值

返回一个列表。

5.8.2 获取列表长度

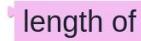
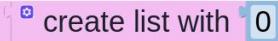
[length of]

- 功能描述

获取列表中元素的个数。

- 用法示例

- **添加列表：** 将要访问的列表拖入到此块中（示例中是临时创建了一个列表）。

- 运行结果及返回值

返回结果为 1。

5.8.3 获取列表元素值

- 功能描述

该块是对列表操作的集合，其中包含了：

获取列表下标的元素的值

获取列表下标的元素的值，并移除该元素

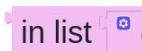
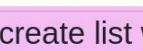
移除列表下标的元素

- 用法示例 1：获取列表下标的元素的值

a. **添加列表：**将要访问的列表拖入到此块中（示例中是临时创建了一个列表）。

b. **选择操作：**选择 get 操作。

c. **添加下标值块：**添加要操作的列表的下标。


- 运行结果及返回值

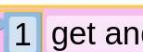
结果返回临时列表中下标为 1 的元素的值，这里的结果为 1。此时，列表长度仍为 2。

- 用法示例 2：获取列表下标的元素的值，并移除该元素

a. 同上

b. **选择操作：**选择 get and remove 操作。

c. 同上


- 运行结果及返回值

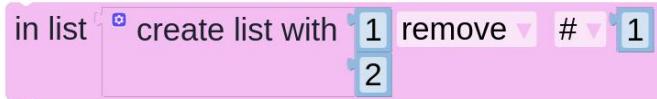
结果返回临时列表中下标为 1 的元素的值，这里的结果为 1。并且，此临时列表的第一个元素 1 已经被移除，列表长度此时为 1。

- 用法示例 3：移除列表下标的元素

a. 同上

b. **选择操作：**选择 remove 操作。

c. 同上



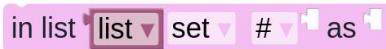
- 运行结果及返回值

不返回任何值，此临时列表的第一个元素 1 已经被移除，列表长度此时为 1。

- 注意事项

- 注意，列表的下标是从 1 开始，而不是从 0 开始。

5.8.4 设置列表元素值



- 功能描述

改变列表下标的元素的值，或者插入值到列表指定下标中。

- 用法示例 1：改变列表下标的元素的值

- a. **添加列表：**将要访问的列表拖入到此块中（示例中是临时创建了一个列表）。
- b. **选择操作：**选择 **set** 操作。
- c. **添加下标值块：**添加要操作的列表的下标。
- d. **添加值块：**添加要设置的目标值块。



- 运行结果及返回值

不返回任何值，此临时列表的第一个元素 1 的值已经变成了 0。原始列表：[1, 2]，修改后的列表：[0, 2]。

- 用法示例 2：插入值到列表指定下标

- a. 同上

- b. **选择操作：**选择 **insert at** 操作。

- c. 同上

- d. 同上



- 运行结果及返回值

不返回任何值，此临时列表下标为 1 的位置已经被插入一个新的值 0。原始列表：[1, 2]，修改后的列表：[0, 1, 2]。

5.8.5 列表和字符串转换

 make list from text  with delimiter 

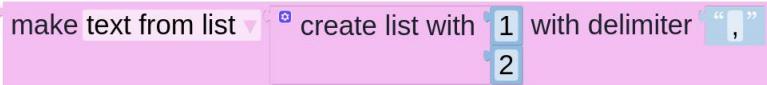
- **功能描述**
 - 将字符串根据分隔符分割，存储在一个列表中。
 - 将列表以指定的分割符为间隔，存储在一个字符串中。
- **用法示例 1：**将字符串根据分隔符分割，存储在一个列表中
 - a. **选择操作：**选择 **list from text** 操作。
 - b. **添加字符串：**将要转换的字符串添加到此块中。
 - c. **指定分隔符：**指定分隔符 `,`。

 make list from text  “1,2,3,4” with delimiter 

- **运行结果及返回值**

返回一个列表，存储着字符串中以 `,` 分隔开的值：[1, 2, 3, 4]。

- **用法示例 2：**将列表根据以分隔符为间隔，存储在一个字符串中
 - a. **选择操作：**选择 **text from list** 操作。
 - b. **添加列表：**将要访问的列表拖入到此块中（示例中是临时创建了一个列表）。
 - c. **指定分隔符：**指定分隔符 `,`。

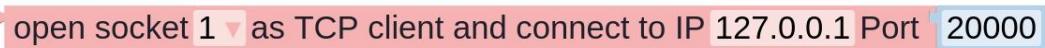
 make text from list  create list with  1 with delimiter 
2

- **运行结果及返回值**

返回一个字符串，存储着列表中的值，以 `,` 分隔开：“1,2”。

5.9 Communication 外部通讯

5.9.1 打开 Socket 连接

 open socket  1 as TCP client and connect to IP **127.0.0.1** Port **20000**

- **功能描述**

此块用于尝试与指定的 IP 地址和端口号建立 TCP 连接，并管理一个 Socket ID。
- **用法示例**
 - a. **选择 Socket ID：**在下拉列表中选择一个 Socket ID（可选范围为 1-5），每个 ID 对应一个独立的 Socket 连接资源。
 - b. **设置 IP 地址和端口：**输入目标服务器的 IP 地址和端口号。这将是连接尝试的目的地。
 - **运行结果及返回值**

成功: 如果在 2 秒内成功建立连接，返回 `true`。

失败: 如果在 2 秒内未能成功建立连接或遇到任何错误，返回 `false`。

- **注意事项**

- 确保所选的 **Socket ID** 没有被其他任务占用。
- 验证 **IP 地址** 和 **端口号** 的有效性以及目标服务器的可达性。

5.9.2 Socket 发送文本信息

socket 1 send text “Hi, flexiv”

- **功能描述**

使用已建立连接 **Socket ID** 发送文本信息。

- **用法示例**

a. **选择 Socket ID:** 在下拉列表中选择一个 **Socket ID**（可选范围为 1-5），每个 ID 对应一个独立的 **Socket** 连接资源。

b. **输入文本:** 在文本框中输入想要发送的消息。

- **运行结果及返回值**

成功: 如果在 2 秒内成功发送全部文本，返回 `true`。

失败: 如果在 2 秒内未能成功发送或遇到任何错误，返回 `false`。

- **注意事项**

确保在发送数据前，所选的 **Socket ID** 对应的连接已成功建立并且仍然处于活动状态。

5.9.3 Socket 接收文本信息

socket 1 receive text

- **功能描述**

使用已建立连接 **Socket ID** 接收文本信息。

- **用法示例**

◦ **选择 Socket ID:** 在下拉列表中选择一个 **Socket ID**（可选范围为 1-5），每个 ID 对应一个独立的 **Socket** 连接资源。

- **运行结果及返回值**

成功: 如果在 2 秒内接收到文本信息，则返回接收到的文本信息。

失败: 如果在 2 秒内未能接收到任何文本信息或遇到任何错误，返回空的文本。

- **注意事项**

◦ 确保在接收数据前，所选的 **Socket ID** 对应的连接已成功建立并且仍然处于活动状态。

◦ 该接收模块是根据换行符 '`\n`' 来判断结尾的，发送方应该在发送的文本末尾加上换行符 '`\n`'。

5.9.4 Socket 是否连接

socket 1 is connected

- 功能描述

查看指定 Socket ID 是否已经连接。

- 用法示例

- 选择 Socket ID:** 在下拉列表中选择一个 Socket ID（可选范围为 1-5），每个 ID 对应一个独立的 Socket 连接资源。

- 运行结果及返回值**

已连接: 如果 Socket ID 已连接，返回 true。

未连接: 如果 Socket ID 未连接，返回 false。

5.9.5 关闭 Socket 连接

close socket 1

- 功能描述

关闭指定 Socket ID 连接。

- 用法示例

- 选择 Socket ID:** 在下拉列表中选择一个 Socket ID（可选范围为 1-5），每个 ID 对应一个独立的 Socket 连接资源。

- 运行结果**

运行此块后，会关闭 Socket ID 为 1 的 Socket 连接。如果该 Socket ID 上在此之前没有建立连接，运行此块也不会报错。

5.9.6 打开 Modbus TCP 连接

open Modbus TCP master ID 1 and connect to IP 127.0.0.1 Port 2000

- 功能描述

此块用于尝试与指定的 IP 地址和端口号建立 Modbus TCP 连接，并管理一个 Modbus TCP ID。

- 用法示例

- a. 选择 ModbusTCP ID:** 在下拉列表中选择一个 Modbus TCP ID（可选范围为 1-5），每个 ID 对应一个独立的 Modbus TCP 连接资源。

- b. 设置 IP 地址和端口:** 输入目标服务器的 IP 地址和端口号。这将是连接尝试的目的地。

- 运行结果及返回值**

成功: 如果在 2 秒内成功建立连接，返回 true。

失败: 如果在 2 秒内未能成功建立连接或遇到任何错误，返回 false。

- 注意事项
 - 确保所选的 Modbus TCP ID 没有被其他任务占用。
 - 验证 IP 地址和端口号的有效性以及目标服务器的可达性。

5.9.7 Modbus TCP 写入寄存器

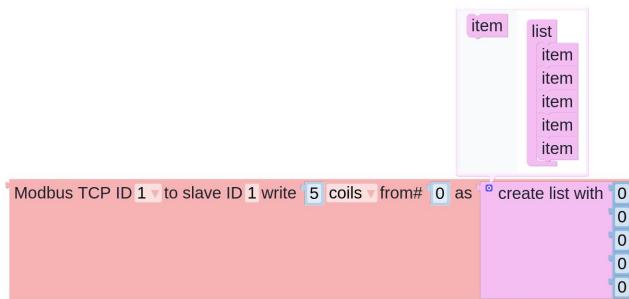
Modbus TCP ID 1 to slave ID 1 write 2 coils from# 0 as create list with 0 0

- 功能描述

使用已建立连接的 Modbus TCP ID，写入数据到从站的寄存器之中。

- 用法示例

- 选择 **Modbus TCP ID**: 在下拉列表中选择一个 Modbus TCP ID (可选范围为 1-5)，每个 ID 对应一个独立的 Modbus TCP 连接资源。
- 输入 Slave ID**: 在输入框中输入目标从站的 Slave ID，结合具体的从站设备配置来进行输入。
- 输入长度**: 在输入框中输入要写入的数据长度。
- 选择寄存器类型**: 在下拉列表中选择要写入的寄存器类型。
- 输入起始地址**: 在输入框中输入要写入的寄存器的起始地址。
- 添加发送数据**: 建立要发送的列表，可以是任意长度的列表，但是长度必须与 c. 输入长度保持一致。比如:



List 中每个元素大小为一个字节的无符号整数 (`uint8_t`)，所以当输入的数字大于 255、小于 0 或者为非整数时，可能会发送一个非预期的结果。所以，请提前确保 List 中每个元素都在一个字节的无符号整数范围内，并且提前处理（转换或者截断）非正整数的值。

- 运行结果及返回值
 - 成功**: 如果在 2 秒内成功写入从站寄存器，返回 `true`。
 - 失败**: 如果在 2 秒内未能写入从站寄存器或遇到任何错误，返回 `false`。
- 注意事项
 - 确保在发送数据前，所选的 Modbus TCP ID 对应的连接已成功建立并且仍然处于活动状态。

5.9.8 Modbus TCP 读取寄存器

Modbus TCP ID 1 ▼ from slave ID 1 read 2 coils ▼ from# 0

- 功能描述

使用已建立连接 Modbus TCP ID，从指定的从站寄存器中读取数据。

- 用法示例

a. 选择 Modbus TCP ID: 在下拉列表中选择一个 Modbus TCP ID (可选范围为 1-5)，每一个 ID 对应一个独立的 Modbus TCP 连接资源。

b. 输入 Slave ID: 在输入框中输入目标从站的 Slave ID，结合具体的从站设备配置来进行输入。

c. 输入长度: 在输入框中输入要读取的数据长度。

d. 选择寄存器类型: 在下拉列表中选择要读取的寄存器类型。

e. 输入起始地址: 在输入框中输入要读取的寄存器的起始地址。

- 运行结果及返回值

成功: 如果在 2 秒内读取成功，则返回目标从站寄存器的列表值。

失败: 如果在 2 秒内未能读取成功或遇到任何错误，返回空的列表。

- 注意事项

◦ 确保在接收数据前，所选的 Modbus TCP ID 对应的连接已成功建立并且仍然处于活动状态。

5.9.9 Modbus TCP 是否连接

Modbus TCP ID 1 ▼ is connected

- 功能描述

查看指定 Modbus TCP ID 是否已经连接。

- 用法示例

◦ 选择 Modbus TCP ID: 在下拉列表中选择一个 Modbus TCP ID (可选范围为 1-5)，每一个 ID 对应一个独立的 Modbus TCP 连接资源。

- 运行结果及返回值

已连接: 如果 Modbus TCP ID 已连接，返回 true。

未连接: 如果 Modbus TCP ID 未连接，返回 false。

5.9.10 关闭 Modbus TCP 连接

close Modbus TCP ID 1 ▾

- 功能描述

关闭指定 Modbus TCP ID 连接。

- 用法示例

- 选择 Modbus TCP ID:** 在下拉列表中选择一个 Modbus TCP ID（可选范围为 1-5），每一个 ID 对应一个独立的 Modbus TCP 串口连接资源。

- 运行结果**

运行此块后，会关闭 Modbus TCP ID 为 1 的 Modbus TCP 连接。如果该 Modbus TCP ID 上在此之前没有建立连接，运行此块也不会报错。

5.9.11 打开 Modbus RTU 连接

open Modbus RTU master ID 1 ▾ with Port /dev/ttyS0 ▾
BaudRate 115200 ▾ Parity None ▾ DataBits 8 ▾ StopBits 1 ▾

- 功能描述

此块用于打开一个 Modbus RTU master，并管理一个 Modbus RTU ID。

- 用法示例

- a. 选择 Modbus RTU ID:** 在下拉列表中选择一个 Modbus RTU ID（可选范围为 1-5），每一个 ID 对应一个独立的 Modbus RTU 连接资源。

- b. 选择串口:** 在下拉列表中选择一个串口名称。

- c. 选择波特率、校验位、数据位、停止位:** 在下拉列表中一一设置相应的串口通信参数。

- 运行结果及返回值**

成功: 如果成功建立连接，返回 true。

失败: 如果遇到任何错误，立即返回 false。

5.9.12 Modbus RTU 写入寄存器

Modbus RTU ID 1 ▾ to slave ID 1 write 2 coils ▾ from# 0 as 1 create list with 0
0

- 功能描述

使用已建立连接的 Modbus RTU ID，写入数据到从站的寄存器之中。

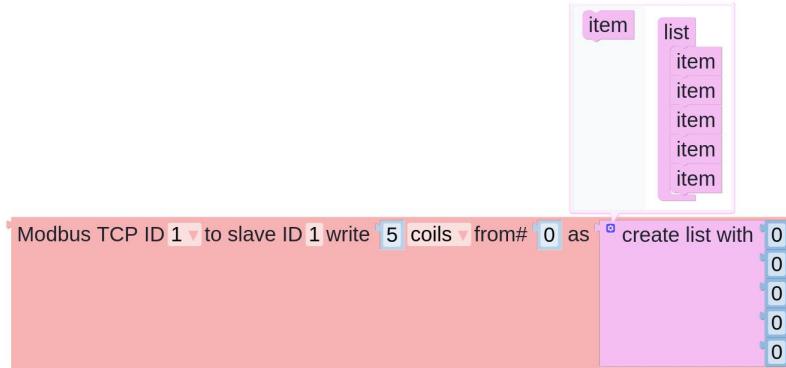
- 用法示例

- a. 选择 Modbus RTU ID:** 在下拉列表中选择一个 Modbus RTU ID（可选范围为 1-5），每一个 ID 对应一个独立的 Modbus RTU 连接资源。

- b. 输入 Slave ID :** 在输入框中输入目标从站的 Slave ID，结合具体的从站设备配置来进行

输入。

- c. **输入长度:** 在输入框中输入要写入的数据长度。
- d. **选择寄存器类型:** 在下拉列表中选择要写入的寄存器类型。
- e. **输入起始地址:** 在输入框中输入要写入的寄存器的起始地址。
- f. **添加发送数据:** 建立要发送的列表，可以是任意长度的列表，但是长度必须与 c. 输入长度 保持一致。比如：



List 中每个元素大小为一个字节的无符号整数 (`uint8_t`)，所以当输入的数字大于 255、小于 0 或者为非整数时，可能会发送一个非预期的结果。所以，请提前确保 List 中每个元素都在一个字节的无符号整数范围内，并且提前处理（转换或者截断）非正整数的值。

- **运行结果及返回值**

成功: 如果在 2 秒内成功写入从站寄存器，返回 `true`。

失败: 如果在 2 秒内未能写入从站寄存器或遇到任何错误，返回 `false`。

5.9.13 Modbus RTU 读取寄存器

Modbus RTU ID 1 ▼ from slave ID 1 read 2 coils ▼ from# 0

- **功能描述**

使用已建立连接 Modbus RTU ID，从指定的从站寄存器中读取数据。

- **用法示例**

- a. **选择 Modbus RTU ID:** 在下拉列表中选择一个 Modbus RTU ID（可选范围为 1-5），每一个 ID 对应一个独立的 Modbus RTU 连接资源。

- b. **输入 Slave ID:** 在输入框中输入目标从站的 Slave ID，结合具体的从站设备配置来进行输入。

- c. **输入长度:** 在输入框中输入要读取的数据长度。

- d. **选择寄存器类型:** 在下拉列表中选择要读取的寄存器类型。

- e. **输入起始地址:** 在输入框中输入要读取的寄存器的起始地址。

- **运行结果及返回值**

成功: 如果在 2 秒内读取成功，则返回目标从站寄存器的列表值。

失败: 如果在 2 秒内未能读取成功或遇到任何错误，返回空的列表。

5.9.14 关闭 Modbus RTU 连接

close Modbus RTU ID 1 ▾

- **功能描述**

关闭指定 Modbus RTU ID 连接。

- **用法示例**

- **选择 Modbus RTU ID:** 在下拉列表中选择一个 Modbus RTU ID（可选范围为 1-5），每个 ID 对应一个独立的 Modbus RTU 串口连接资源。

- **运行结果**

运行此块后，会关闭 Modbus RTU ID 为 1 的 Modbus RTU 连接。如果该 Modbus RTU ID 上在此之前没有建立连接，运行此块也不会报错。

5.9.15 打开 RS485 连接

open RS485 1 ▾ with Port /dev/ttyS0 ▾
BaudRate 115200 ▾ Parity None ▾ DataBits 8 ▾ StopBits 1 ▾

- **功能描述**

此块用于打开一个 RS485 串口通信连接，并管理一个 RS485 ID。

- **用法示例**

- **选择 RS485 ID:** 在下拉列表中选择一个 RS485 ID（可选范围为 1-5），每个 ID 对应一个独立的 RS485 连接资源。

- **选择串口:** 在下拉列表中选择一个串口名称。

- **选择波特率、校验位、数据位、停止位:** 在下拉列表中一一设置相应的串口通信参数。

- **运行结果及返回值**

成功: 如果成功建立连接，返回 true。

失败: 如果遇到任何错误，立即返回 false。

5.9.16 RS485 发送数据

RS485 1 ▾ send list ▾ create list with 0
0

- **功能描述**

使用已建立连接 RS485 ID 发送数据。

- **用法示例**

- **选择 RS485 ID:** 在下拉列表中选择一个 RS485 ID（可选范围为 1-5），每个 ID 对应一

个独立的 RS485 连接资源。

- b. **添加发送数据:** 建立要发送的列表，可以是任意长度、任意形式的列表，比如：



List 中每个元素大小为一个字节的无符号整数 (`uint8_t`)，所以当输入的数字大于 255、小于 0 或者为非整数时，可能会发送一个非预期的结果。所以，请提前确保 List 中每个元素都在一个字节的无符号整数范围内，并且提前处理（转换或者截断）非正整数的值。

- **运行结果及返回值**

成功: 如果在 2 秒内成功发送全部文本，返回 `true`。

失败: 如果在 2 秒未能成功发送或遇到任何错误，返回 `false`。

- **注意事项**

确保在发送数据前，所选的 RS485 ID 对应的连接已成功建立并且仍然处于活动状态。

5.9.17 RS485 接收数据

RS485 ID 1 receive list of length 2

- **功能描述**

使用已建立连接 RS485 ID 接收数据。

- **用法示例**

- a. **选择 RS485 ID:** 在下拉列表中选择一个 RS485 ID（可选范围为 1-5），每个 ID 对应一个独立的 RS485 连接资源。

- b. **数据长度:** 填写要一次性接收的数据长度。

- **运行结果及返回值**

成功: 如果在 2 秒内接收到的指定长度的数据，则返回接收到的全部数据。

失败: 如果在 2 秒内未能接收到指定长度的数据，会返回已接收到的部分数据。在阻塞等待数据期间，若遇到任何错误，会立即返回已接收到的数据。

- **注意事项**

确保在接收数据前，所选的 RS485 ID 对应的连接已成功建立并且仍然处于活动状态。

5.9.18 关闭 RS485 连接

close RS485 ID 1 ▾

- 功能描述

关闭指定 RS485 ID 连接。

- 用法示例

- **选择 RS485 ID:** 在下拉列表中选择一个 RS485 ID（可选范围为 1-5），每个 ID 对应一个独立的 RS485 串口连接资源。

- **运行结果**

运行此块后，会关闭 RS485 ID 为 1 的连接。如果该 RS485 ID 上在此之前没有建立连接，运行此块也不会报错。

5.10 Variables 变量管理

5.10.1 创建并行程序变量

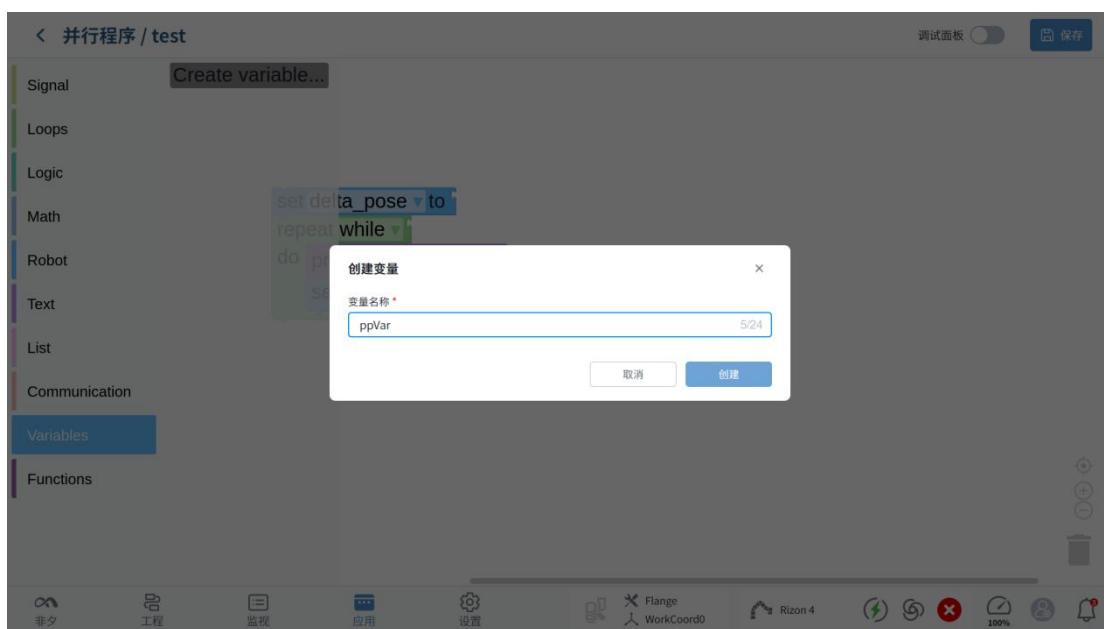
Create variable...

- 功能描述

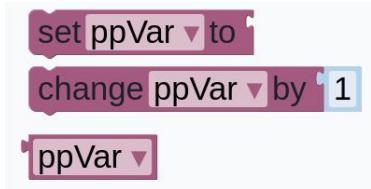
创建一个并行程序变量，该变量只能在此并行程序中使用。

- 用法示例

- 点击 **创建变量**，输入变量名，点击 **创建**。



- 变量创建完成后，会新增三个指令块。



5.10.2 设置变量值

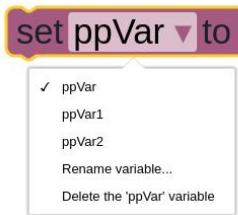
set ppVar to

- 功能描述

给一个并行程序变量赋值。

- 用法示例

a. 选择变量：选择要赋值的并行程序变量的名称。



b. 添加值块：添加一个有返回值的块为并行程序变量赋值，这个值可以是任意类型。



- 运行结果

并行程序变量 ppVar 被依次赋值为：0、true、“Hello Flexiv!”、{0, true, “Hello Flexiv!”}。

5.10.3 变量自增

change ppVar by 1

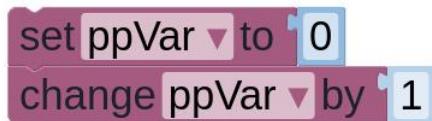
- 功能描述

为并行程序变量自增一个数值。

- 用法示例

a. 选择变量：选择一个并行程序变量的名称，该变量需要首先被赋值为数值类型。

b. 添加值块：添加一个要加上的数值块。



- 运行结果

并行程序变量 ppVar 的值变为 1。

5.10.4 获取变量值



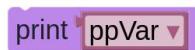
- 功能描述

读取一个并行程序变量的值。

- 用法示例

a. 选择全局变量：选择要读取的并行程序的名称。

b. 使用指令块：将指令块与其他指令块拼接。

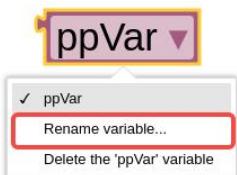


- 运行结果

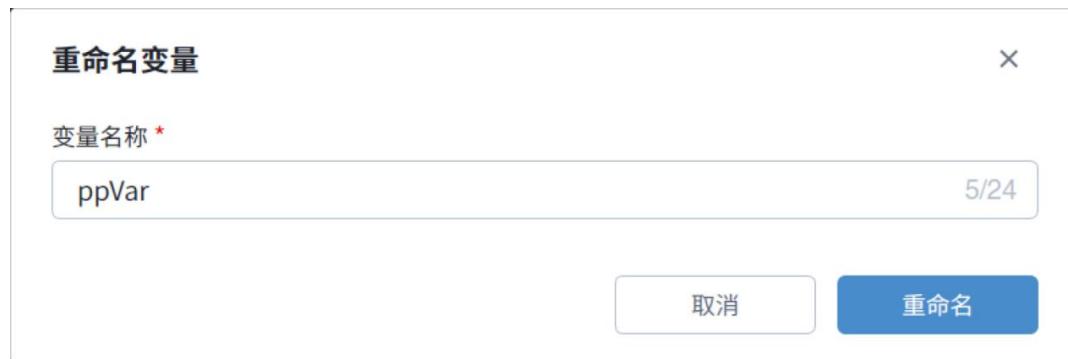
打印出并行程序变量 ppVar 的值。

5.10.5 修改变量名

- 在使用并行程序变量的指令块上，点击并行程序变量名，点击 重命名变量。

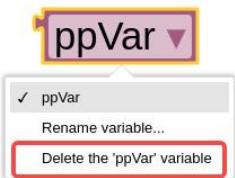


- 在弹出的窗口中输入新的变量名，点击 重命名，即可修改并行程序变量的名称。



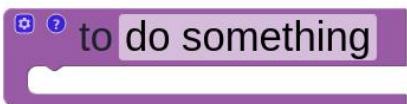
5.10.6 删除变量

- 在使用并行程序变量的指令块上，点击并行程序变量名，点击 **删除变量**，即可将并行程序变量删除。



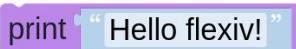
5.11 Functions 函数

5.11.1 定义无返回值函数



- 功能描述

该功能块用于设置一个函数。前面所有的功能块都可以看做是基本语句块，即只能用一次。比如设置完成一个 `print` 的功能块。



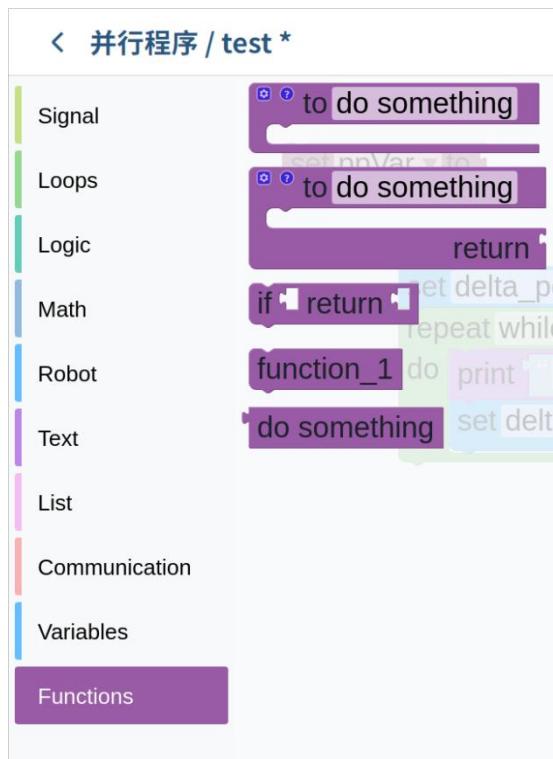
下次还要用的时候，还需要手动设置文本内容“Hello flexiv！”，缺少复用性。为了解决这个问题，可以定义一个函数。

- 用法示例 1

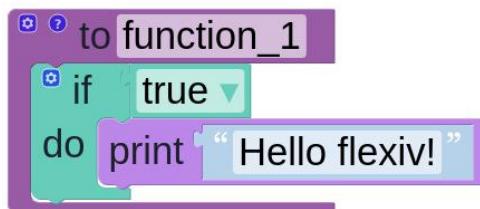
- 设置函数名



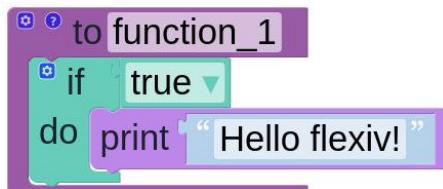
- 设置完函数名，这个函数其实就被定义完成了。在这里可以看到多出了对应函数名的功能块。



c. 添加函数内容。函数内容是任意形式的，可以包含任意逻辑块。这里做一个简单示例：



d. 调用该函数。



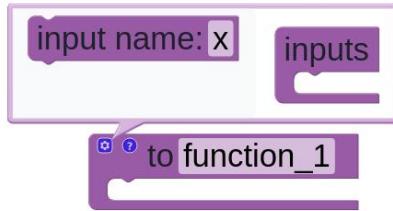
function_1

- 运行结果及返回值

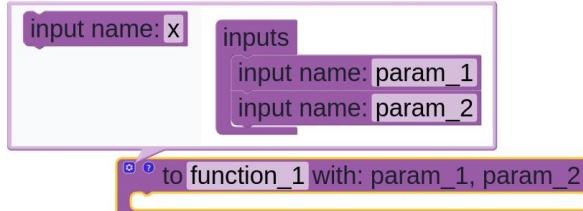
运行后会输出“Hello flexiv!”。

- 用法示例 2

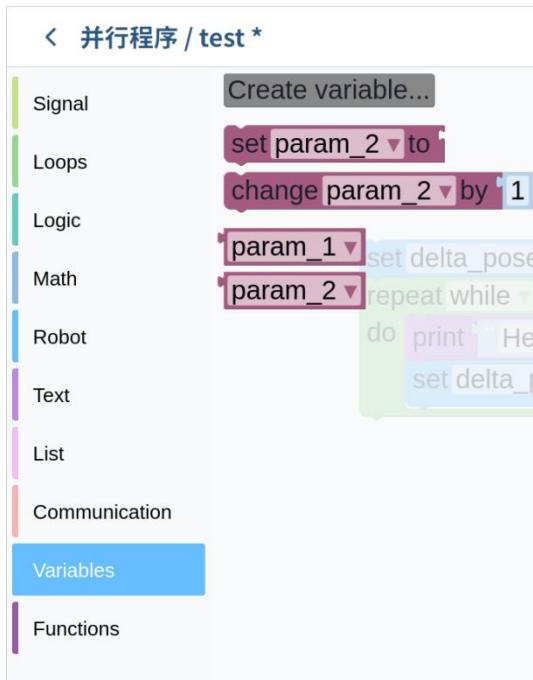
a. 通过设置按钮还可以添加任意数量的参数。



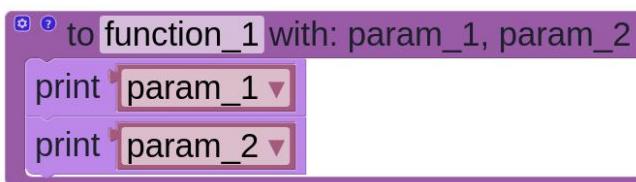
- b. 将 `input name: []` 块拖入 `inputs` 下面即可，并且在输入框内输入参数名。



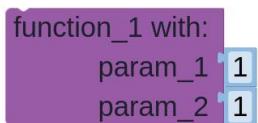
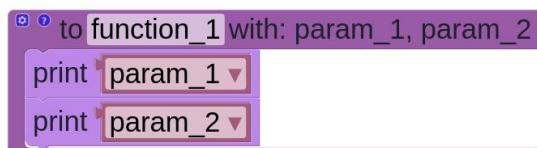
- c. 刚才添加的两个参数会出现在 **变量** 栏中。



- d. 接下来使用使用这两个参数。



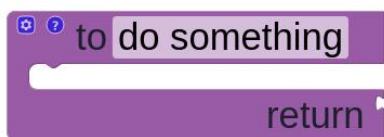
- e. 使用该函数。



- 运行结果及返回值

运行后会输出“1”和“2”。

5.11.2 定义有返回值函数



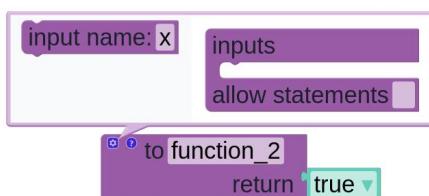
- 功能描述

定义一个函数，并且返回一个值。

- 用法示例

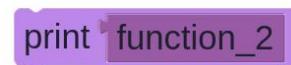
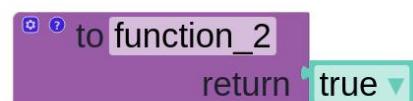
a. 前面的步骤和上一个功能块一致。

b. 在功能块设置中，可以选择是否 allow statements。当为否的时候，该函数就是一个只有返回值的空函数。



c. 添加一个返回结果，比如 true。

d. 使用该函数。



- 运行结果及返回值

该函数返回一个 true。

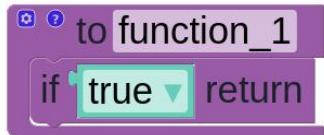
5.11.3 条件返回

- 功能描述

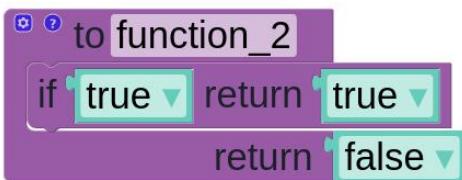
当条件成立，则返回值。

- 用法示例

- 该功能块需要配合前面两个功能块使用。
- 设置成立条件，比如 `true`。因为第一种函数功能块是没有返回值的，所以当当前逻辑块配合它使用的时候，会自动隐藏 `return` 的结果。



- 配合有返回值函数功能块使用时，则可以再设置一个返回值。



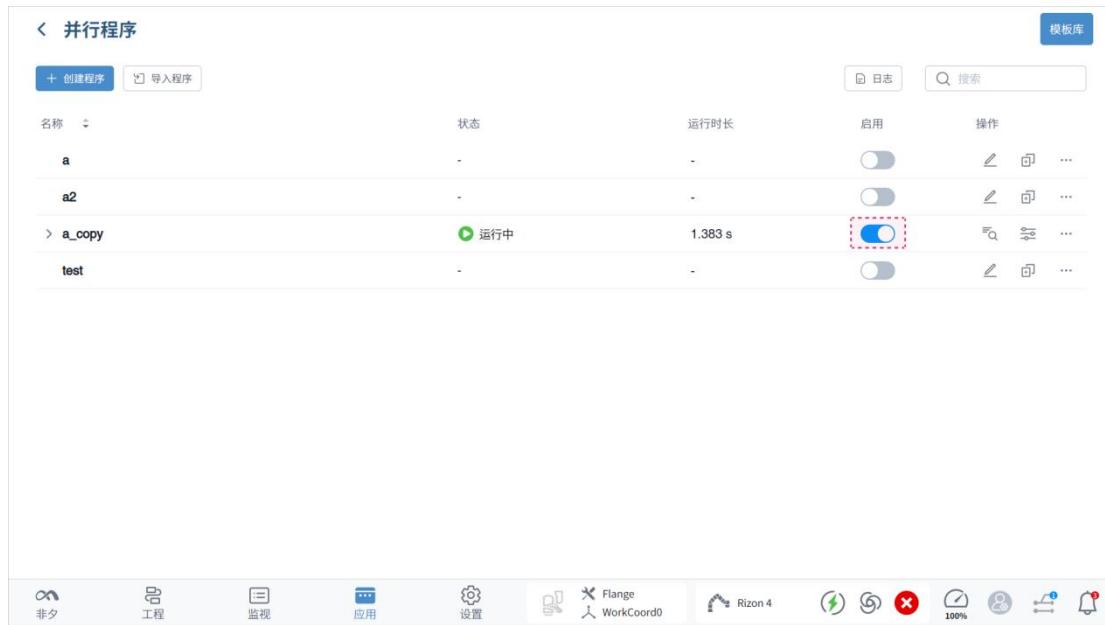
6. 运行并行程序

6.1 启用/停用并行程序

编写完并行程序后，可以在并行程序列表中打开 **启用** 开关，启用并运行相应的并行程序。

对于已经启用的并行程序，您可以关闭 **启用** 开关，停用该并行程序。

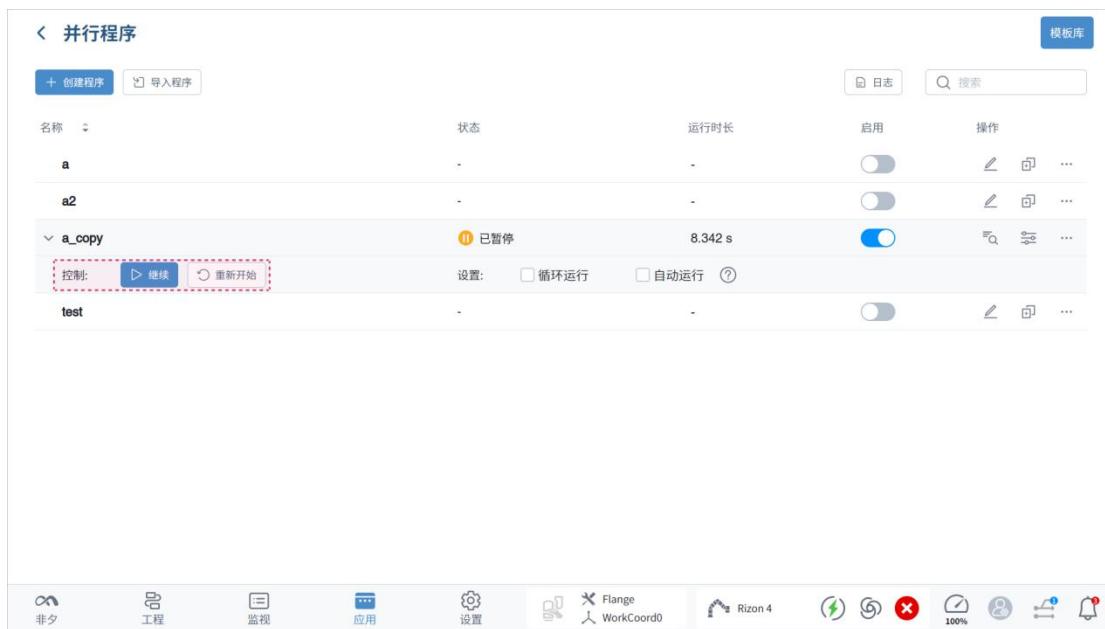
 **注：**一次最多只能启用 5 个并行程序。



名称	状态	运行时长	启用	操作		
a	-	-	<input type="checkbox"/>			
a2	-	-	<input type="checkbox"/>			
> a_copy	运行中	1.383 s	<input checked="" type="checkbox"/>			
test	-	-	<input type="checkbox"/>			

6.2 暂停/继续运行并行程序

点击并行程序名称左侧的 **展开** 图标以展开显示并行程序控制行，点击控制区域的按钮，可以 **暂停**、**继续**、**重新开始** 并行程序。

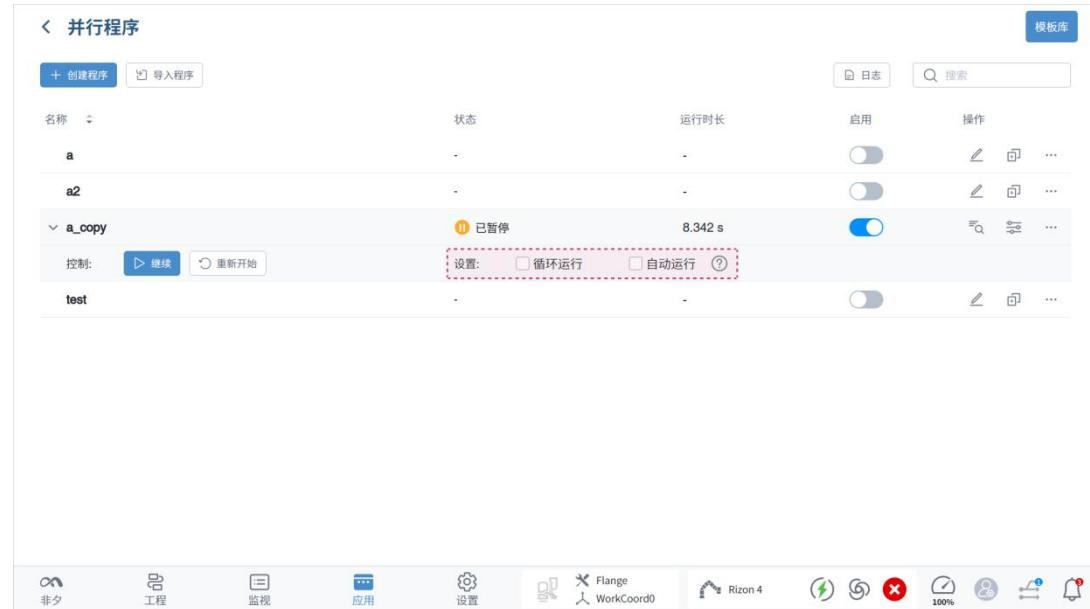


名称	状态	运行时长	启用	操作		
a	-	-	<input type="checkbox"/>			
a2	-	-	<input type="checkbox"/>			
> a_copy	已暂停	8.342 s	<input checked="" type="checkbox"/>			
test	-	-	<input type="checkbox"/>			

6.3 设置运行模式

在并行程序控制行中，可以设置并行程序的运行模式：

- **循环运行：** 并行程序运行结束后自动重新开始运行
- **自动运行：** 机器人开机时自动运行并行程序



7. 查看并行程序状态

7.1 查看并行程序运行状态

已经启用的并行程序会显示其运行状态和运行时间，运行状态包括：

- **运行中：**并行程序正在运行中
- **已暂停：**并行程序已暂停
- **已终止：**并行程序已终止运行
- **错误：**并行程序运行发生错误

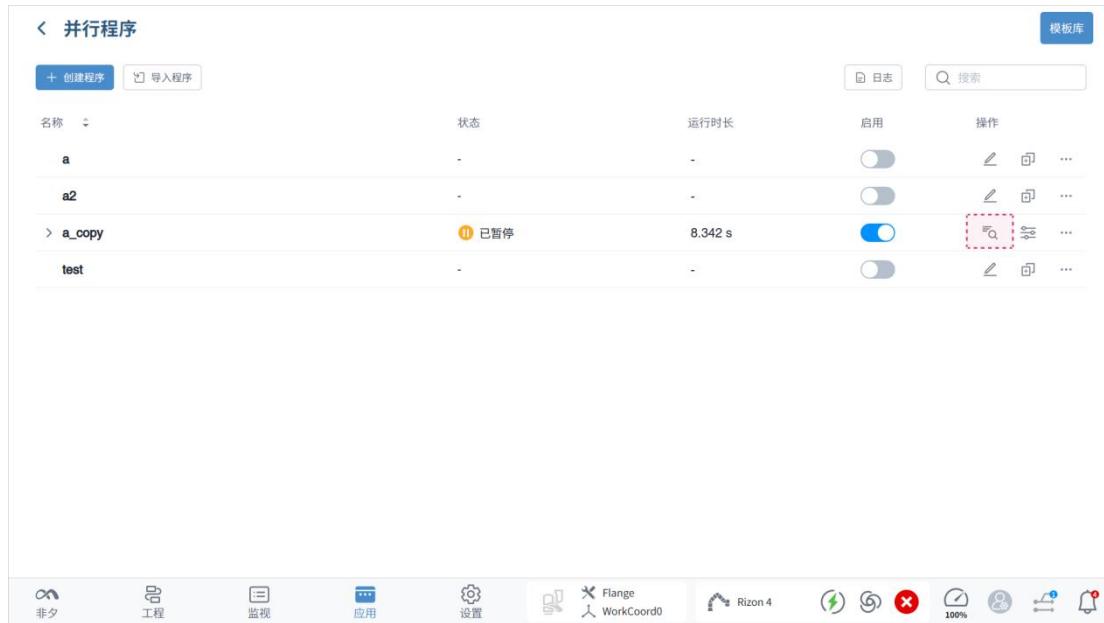
The screenshot shows the 'Parallel Programs' section of the FLEXIV software. At the top, there are buttons for 'Create Program' (+), 'Import Program' (import icon), 'Logs' (log icon), 'Search' (search icon), and 'Template Library' (blue button). Below this is a table listing four parallel programs:

名称	状态	运行时长	启用	操作
> a	运行中	3.251 s	开关图标	日志 搜索 ...
> a2	已终止	0.002 s	开关图标	日志 搜索 ...
> a_copy	已暂停	8.342 s	开关图标	日志 搜索 ...
> test	错误	0.000 s	开关图标	日志 搜索 ...

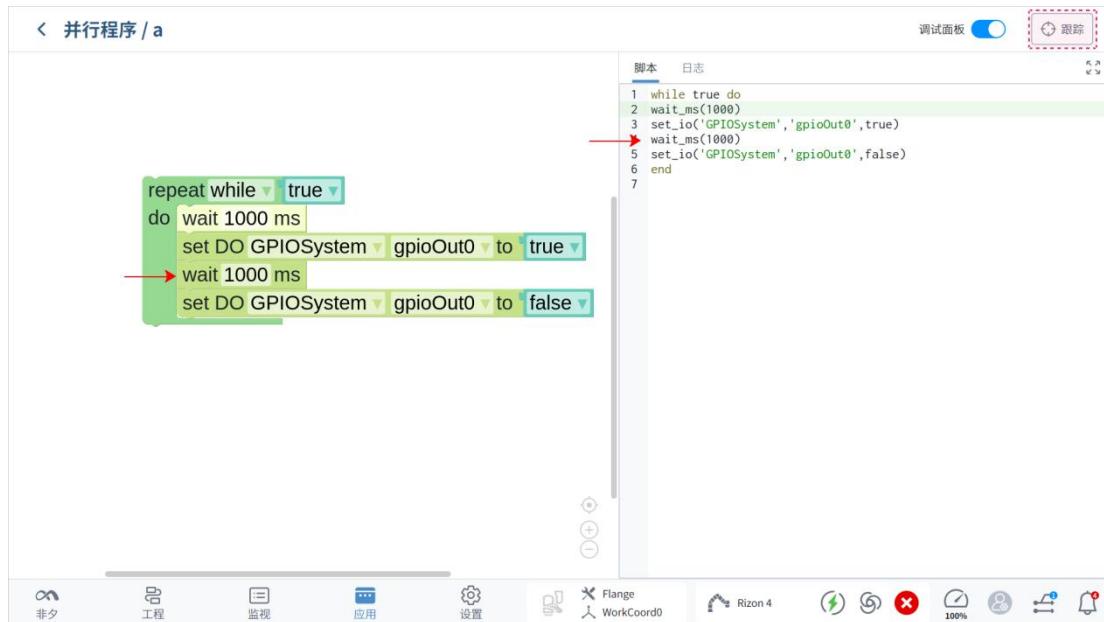
At the bottom of the interface, there is a navigation bar with icons for 'Home' (非夕), 'Project' (工程), 'Monitor' (监视), 'Applications' (应用), 'Settings' (设置), 'Flange' (Flange), 'WorkCoord' (WorkCoord), 'Rizon 4' (Rizon 4), and various status indicators like battery level (100%), signal strength, and connectivity.

7.2 查看并行程序运行详情

点击已启用的并行程序右侧的 **查看** 图标，可以进入并行程序运行界面查看并行程序的详细信息。



在并行程序运行界面，会高亮显示正在运行的指令块。同时，打开调试面板，运行中的脚本指令也会高亮显示。点击 **跟踪**，运行中的指令块将居中显示。



7.3 查看并行程序日志

点击 **日志** 可查看当前运行的所有并行程序的日志。

The screenshot shows the 'Parallel Programs' section of the Flexiv Elements software. At the top right, there is a 'Logs' button with a red dashed border. Below it is a search bar with the placeholder 'Search'. The main area displays a table with columns: '名称' (Name), '状态' (Status), '运行时长' (Run Time), '启用' (Enabled), and '操作' (Operations). Five parallel programs are listed: 'a', 'a2', 'a_copy', 'test', and another unnamed entry. Each row has a 'Logs' button in the '操作' column. The bottom of the screen features a navigation bar with icons for '首页' (Home), '工程' (Project), '监视' (Monitor), '应用' (Application), '设置' (Settings), and several status indicators like 'Flange WorkCoord0', 'Rizon 4', and battery levels.

7.4 查看导航栏状态

Flexiv Elements 底部导航栏的并行程序状态区域会显示当前并行程序运行状态。若未启用并行程序，将不会显示该状态。

This screenshot shows the same 'Parallel Programs' section as the previous one, but with a different state for the 'a_copy' program. The 'a_copy' row now has a green circular icon with a play symbol and the text '运行中' (Running) next to it. Its '启用' (Enabled) switch is also highlighted with a red dashed border. The other programs ('a', 'a2', 'test') remain in their previous states. The bottom navigation bar includes a '暂停' (Pause) button next to the '控制' (Control) label, which is highlighted with a red box. The rest of the interface and status indicators at the bottom are identical to the first screenshot.

并行程序状态	描述	显示
已启用，运行正常	并行程序已启用且运行正常 同时显示并行程序的启用数量	
已启用，发生错误	并行程序已启用，且有并行程序发生错误 同时显示并行程序的启用数量	
未启用	未启用任何并行程序	无

点击该区域可查看并行程序运行的运行状态和运行时间。点击 [查看详情](#) 可跳转至 [应用 > 并行程序](#) 界面查看或编辑并行程序。点击 [日志](#) 可查看并行程序的日志。





并行程序用户手册